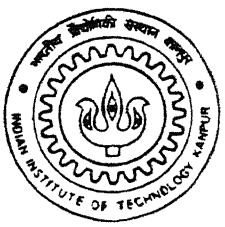
BOUNDARY DETECTION IN 3D MRI IMAGES USING LEVEL SET METHODS

by

Ritesh Kumar Pila



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

March, 1999

Boundary Detection in 3D MRI Images Using Level Set Methods

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

by

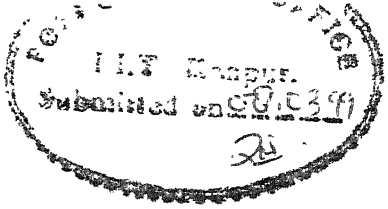
RITESH KUMAR PILA

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

March, 1999



Certificate

It is certified that the work contained in the thesis entitled BOUNDARY DETECTION IN 3D MRI IMAGES USING LEVEL SET METHODS, by Ritesh Kumar Pila, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. G. Sharma)

Professor

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

March, 1999.

To

My Parents

Acknowledgements

I would like to express my sincere gratitude to Dr. G. Sharma for his invaluable guidance and constant encouragement in completing my thesis work. I am also grateful to Dr. A.K. Raina for his moral support and kind cooperation.

I express my sincere respect for Shafi who has helped, guided and encouraged me like a brother during my thesis semester. I also wish to thank Udayji and Ramnathji for their extended helping hand in Control System Lab.

I would love to mention the names of my friends Anirban(banu), Ashok(ji), Gautamda, Gopal, Himadri(himu), Jalda, Karthik(Y'M), Mashuq, Pratul , Prashant, Pratibha, Samarjeet, Siddharth, Mama who have made my stay at IITK, an unforgettable experience. I will always cherish the time spent with them.

I wish to thank specially Arindam(antu), Jay(mota), Sandhitsu(sandy) who have helped me by sharing their valuable ideas and suggestions in my thesis work.

I must not forget to mention the name Shiva who was always beside me and has made the IITK moments the most memorable one of my life.

Last but not the least I wish to acknowledge the constant support, blessings and guidance which my parents gave me.

Abstract

It is well known that in Digital Imaging, detection of boundaries both in 2D and 3D is very important. In medical imaging one is interested only in a specific region of image, so one needs an algorithm for isolating accurately the region of interest. There are several algorithms available with their own merits and demerits. In this thesis we have used the concept of active contour method using the level set. By this method we move the initial contour till it hits the boundary of the object. To achieve the efficiency in computation we have used a narrow banding method by which only a portion of the image plane is considered for the calculation. In narrow band method we rebuild the level set function whenever it collides with the boundary and use it as zero level set(initial front) for further evolution. We have presented a number of MRI images with the segmented objects of interest. Examples have been given in both 2D and 3D. We have used the Visualization Tool Kit for displaying 3D images. In our method, objects of interest are chosen by the user by specifying an initial curve or surface.

Contents

List of Figures

Chapter 1

Introduction

One of the primary problems in image analysis is image segmentation. In image segmentation the basic job is to divide image into regions of similar properties. In many applications one needs to know the required object-boundaries accurately.

1.1 Classical Techniques

Since segmentation is the first building block for all further image processing of medical images, there are many techniques to solve this problem. In image segmentation, image can be divided into it's components based on abrupt changes in grey level of pixels. In medical imaging one of the main objective of segmentation is to isolate a region of interest and accurately determine its boundary. Simple segmentation schemes do not perform satisfactorily. For isolating a particular object of interest, several techniques, such as region growing, region splitting and merging and thresholding the region techniques can be used [2]. All these techniques of segmentation are based on grey level value similarity and can be used for both static and dynamic images (i.e. images varying with time).

Thresholding is a basic technique which is found to be very useful in many segmentation techniques particularly in infrared imaging in military applications. Here the

targets of interest are normally hotter than background. Thus they appear brighter than the background. So if the threshold of the image can be set to a particular suitable value, the objects of interest can be easily isolated from the rest of the background. But such simple techniques are not useful for medical images.

Region growing, splitting and merging techniques work better in medical imaging application. Normally in these techniques we have to give an initial point(seed) which lies inside the region of interest. In image growing process the simplest approach is pixel aggregation which starts with sets of seed points(user specified) and these grow into regions by appending to each seed point neighboring pixels that have similar properties(user specified). One of the problems in region growing technique is the formulation of a stopping rule. Region growing should stop when pixels no longer satisfy the criteria for inclusion in that region. Criteria such as intensity, texture and color are local to the image and do not take into account the history of region's growth. Some of these properties of image can be used for formulation of stopping criteria but no satisfactory rule has been found. In region splitting and merging technique if each pixel or some percentage of pixels(user specified) do not have the similar properties then the whole region is split into a number of square blocks. Next the subregions having similarities are merged. After this merging operation again the subregions which is satisfying the splitting rules is split and the whole process is repeated till no further splitting is possible. Apart from these classical methods there are some recently developed methods in which better results and more flexibilities have been achieved. Some of these methods are described below.

1.2 Some Recent Techniques

1.2.1 Segmentation Using Fuzzy Clustering Technique

One of the recent technique is segmentation using Fuzzy clustering method in which by clustering the input image data, the boundaries of different objects can be found out [4]. Normally Fuzzy c-means clustering is used to cluster each slice from a volume data set into a set of classes. The membership information within a class is used as a guide to re-cluster the areas of the selected region. This improves the accuracy of the detected boundary.

1.2.2 Automatic Segmentation by Rule Based Expert System

Another technique is to use knowledge based technique in which the segmented image of some unsupervised clustering algorithm along with cluster centers are provided to a rule-based expert system which then extracts the required regions [5]. The unsupervised clustering can be performed by Fuzzy c-means clustering as mentioned in the above subsection and by any other method.

1.2.3 Segmentation Using Active Contour Model (Snakes)

In the recent past one of the solutions to this boundary detection problem was given by Kass et al [1]. They proposed the model of snakes or active contours which is based on deforming an initial contour or surface towards the boundary of the objects to be detected. The deformation is obtained by minimizing the energy functional such that its minimum is obtained at the boundary of the object. The energy functional is chosen such that it's local minima comprise the set of alternative solutions. The choice among these alternatives could require some type of minimization technique or

high-level search. Also interactive approach can be used to explore the alternatives. By adding suitable energy terms to the minimization, it is possible for a user to push the model out of a local minimum toward the global one which in fact is the desired solution. The result is an active contour that falls into the desired solution when placed sufficiently near the optimum. One of the disadvantage is that this curve is not capable of changing its topology. The topology of the final curve/surface will be same as that of the initial curve/surface. This is certainly a problem when unknown number of objects must be detected simultaneously. This snake model can be generalized to 3D images where the boundaries are surfaces. We have discussed this model in details as follows.

Let $C(p) = (x(p), y(p))^T$ be a closed contour in R^2 where $0 \leq p \leq 1$. The subscript T is for the transpose. We now define an energy functional on the set of each contours ("snakes"), $\mathcal{E}(C)$. Following standard practice, we take $\mathcal{E}(C)$ to be of the form

$$\mathcal{E}(C) = \mathcal{E}_{int}(C) + \mathcal{P}(C)$$

where $\mathcal{E}_{int}$ is the *internal deformation energy* and $\mathcal{P}$ is an external potential energy which depends on the image. Other external constraint forces may be added to it. Perhaps the most common choice for the internal energy is the quadratic functional

$$\mathcal{E}_{int} := \int_0^1 (w_1(p)||C_p||^2 + w_2(p)||C_{pp}||^2)dp \tag{1.1}$$

where w_1 and w_2 controls the "tension" and "rigidity" of the snake, respectively.

Let us consider a grey-scale image $I(x, y)$. Then the *external potential energy* depends on the image $I(x, y)$. It can be defined by

$$\mathcal{P}(C) := \int_0^1 P(C(p))dp \tag{1.2}$$

where $P(x, y)$ is a scalar potential function defined on the image plane. The local minima of P attract the snake. For example, we may choose P to be

$$P(x, y) := c||\nabla G_\sigma * I(x, y)|| \tag{1.3}$$

for a suitably chosen constant c, in which case the snake will be attracted towards the intensity edges. Here G_σ denotes a Gaussian smoothing filter of standard deviation σ.

Solving the problem of snakes amounts to finding, for a given set of weights w_1, w_2, the curve C that minimizes $\mathcal{E}$. The classical snake-method provides an accurate location of the edges sufficiently near a given initialization of the curve, it has ability to extract smooth shapes, can retrieve angles. On the other hand it does not directly allows simultaneous treatment of multiple contours. The classical energy approach of snakes can not deal with changes in topology, unless special topology handling procedures are added. This is the basic formulation of two-dimensional active contours.

One can also typically consider dynamic time-varying models in which $C(p)$ becomes a function of time as well. In this case, one defines a kinetic energy and the corresponding Lagrangian (the difference between the kinetic energy and the energy $\mathcal{E}$ defined above). Applying the principle of least actions, one derives the corresponding Lagrangian equation which one tries to solve numerically employing various approximations.

1.3 Our Methodology

We have implemented a different approach to object detection, by using the active contour (snake) as the zero level set of a higher dimensional function. While this function always stays smooth, it's zero level set may take any possible shape, including sharp corners and also changes in topology. This ability is extremely useful in many applications where the shape is not smooth or the number of objects to be detected are unknown or vary in an image sequence.

The application for the presented technique are numerous. Many problems in medical and other image processing applications can be solved which could not be

or would take much more effort with classical methods. An important concern in object tracking and motion detection application is topological change resulting from tracking positions of object boundaries in an image sequence through time. During their evolution, these closed contours may change connectivity and split, thereby undergoing a topological transformation. One such example is the splitting of cell boundary in a sequence of images depicting cell division. Level set method has been described in Chapter 2.

1.4 Visualization

Visualization is also an important part of the whole process. There are several visualization software available in different platforms now-a-days. We have used visualization toolkit (VTK) for our purpose of 3D visualization. For visualization in 2D we have used x-view (XV). VTK allows users to program according to the user's requirement. As VTK supports Tcl/Tk so user interfaces can be easily developed which otherwise are difficult in Motif or other X-window programming environment. It also supports volume rendering and surface rendering for volume data.

1.5 Organization of The Thesis

Rest of thesis is organized as follows. Chapter 2 describes Level set theory in detail. Chapter 3 comprises the steps of the implementation of Level Set for shape recovery in medical imaging. In Chapter 4 we present the results with discussions. It also concludes the thesis and suggests some future work on this topic.

Chapter 2

Level Set Theories

The material of this chapter is based on *Level Set Methods* by Sethian [3].

2.1 Theory of Front Evolution

Consider a boundary, either a curve in two dimensions or an orientable surface in three dimensions, separating one region from another. Suppose this curve/surface moves in a direction normal to itself (where the normal direction is oriented with respect to an inside or an outside) with a known speed function F. The goal is to track the motion of this interface as it evolves. It is more important to consider the motion in a normal direction rather than in tangential direction in many applications of medical imaging. The speed function F, which may depend on many factors, can be written as:

$$F = F(L, G, I)$$

where

- $L =$ *Local properties* are those, that are determined by local geometric information, such as curvature and normal direction.

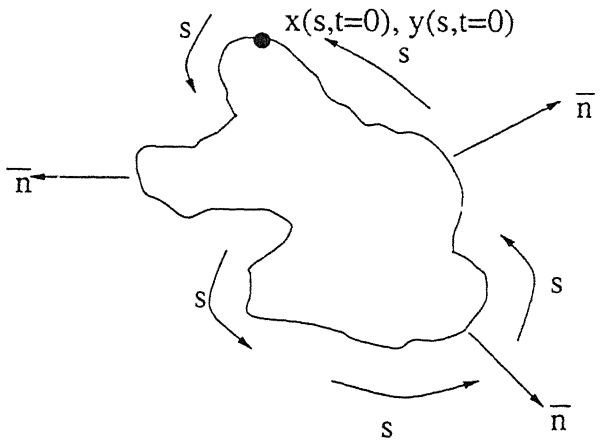


Figure 2.1: Parameterized view of propagating curve

the front and/or associated differential equations. As a particular case, if the interface is a source of heat that affects diffusion on either side of the interface, and a jump in the diffusion in turn influences the motion of the interface, then this would be characterized as front based argument.

- *I = Independent properties* are those that are independent of the shape of the front, such as underlying fluid velocity that passively transports the front.

Let Γ be a simple, smooth, closed initial curve in R^2, and let $\Gamma(t)$ be the one-parameter family of curves generated by moving Γ along its normal vector field with speed F. Here, F is the given scalar function. Thus $\vec{n} \cdot \vec{x_t} = F$, where $\vec{x}$ is the position vector of the curve, t is time and $\vec{n}$ is the unit normal vector to the curve.

A natural approach is to consider a parameterized form of the equations. In this discussion, we will restrict the speed function F to depend only on the local curvature κ of the curve, that is, $F = F(\kappa)$. Curvature is a vector that points in the direction normal to the curve. Let the position vector $\vec{x}(s,t)$ parameterize Γ at time t, where $0 \leq s \leq S$ and S is the total length of the curve at time t. Assume periodic boundary conditions $\vec{x}(0,t) = \vec{x}(S,t)$. The curve is parameterized so that the interior of the closed curve is on the left, in the direction of increasing s as shown in the

Figure 2.1. Let $\vec{n}(s,t)$ be the parameterization of the outward normal and $\kappa(s,t)$ be the parameterization of the curvature. The equations of motion can then be written in terms of individual components $\vec{x} = (x, y)$ as

$$\begin{aligned} x_t &= F\left[\frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2+y_s^2)^{\frac{3}{2}}}\right]\left(\frac{y_s}{(x_s^2+y_s^2)^{\frac{1}{2}}}\right) \\ y_t &= -F\left[\frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2+y_s^2)^{\frac{3}{2}}}\right]\left(\frac{x_s}{(x_s^2+y_s^2)^{\frac{1}{2}}}\right) \end{aligned} \tag{2.1}$$

where the parameterization expression $\kappa = \frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2+y_s^2)^{\frac{3}{2}}}$ is used for the curvature inside the speed function $F(\kappa)$. This is a Lagrangian representation because the range of $(x(s,t), y(s,t))$ describes the moving front.

2.2 Level Set Methods

Given a closed $(N-1)$ dimensional hypersurface $\Gamma(t=0)$, the motion of hyper surface $\Gamma(t)$ propagating along it's normal direction with speed F can be formulated where F can be a function of arguments such as curvature, normal, direction etc.The main idea of the Level Set methodology is to embed this propagating interface as the zero Level Set of a higher dimensional function Ψ. Let $\Psi(x, t=0)$, where x is a point in R^N be defined by

$$\Psi(x, t=0) = \pm d, \tag{2.2}$$

where d is the distance from x to $\Gamma(t=0)$ and the plus (minus) sign is chosen if the point x is outside (inside) the initial hypersurface $\Gamma(t=0)$. Thus we have an initial function

$$\Psi(x, t=0) : R^N \to R,$$

with a property that

$$\Psi(t=0) = [x|\Psi(x, t=0) = 0].$$

with a property that

$$\Psi(t=0) = [x|\Psi(x,t=0) = 0].$$

So at t=0, all the points in the hypersurface those are at zero distance from the initial function i.e. $\Psi(x,t=0)$ comprises of $\Gamma(t=0)$. The goal is to produce an equation for the evolving function $\Psi(x,t)$ that contains the embedded function of $\Gamma(t)$ as the level set $\Psi = 0$. Let $x(t)$ be the path of a point on the propagating front i.e. $x(t=0)$ is a point on the initial front $\Gamma(t=0)$ and $x_t \cdot n = F(x(t))$ with the vector x_t normal to the front $x(t)$. The stipulation that the zero level set of the evolving function Ψ always match the propagating hypersurface means that

$$\Psi(x(t),t) = 0.$$

By the chain rule,

$$\Psi_t + \nabla\Psi(x(t),t) \cdot x_t(t) = 0. \tag{2.3}$$

Since F is the speed in the outward normal direction,

$$\Rightarrow x'(t) \cdot n = F,$$

where $n = \frac{\nabla\Psi}{|\nabla\Psi|}$.

This yields an evolution equation for Ψ i.e.

$$\Rightarrow \Psi_t + F|\nabla\Psi| = 0, \tag{2.4}$$

given $\Psi(x,t=0)$.

This is the Level Set equation introduced by Osher and Sethian [6].

In Figure 2.2(a), an initial circle is shown, together with the circle at a later time in Figure 2.2(c). Figure 2.2(b) shows the associated initial position of the level set function Ψ and Figure 2.2(d) this function at a later time.This formulation is known as an Eulerian Hamilton Jacobi formulation.

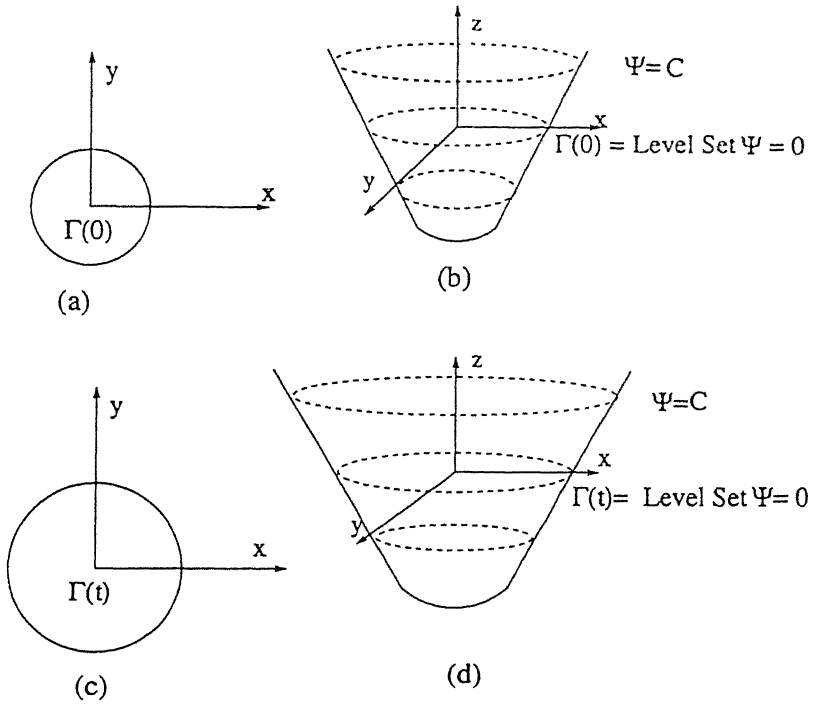


Figure 2.2: Propagating circle

2.2.1 Aspects of the Level Set Formulation

There are several desirable aspects of this formulation.

- The evolving function $\Psi(x,t)$ always remains a function as long as F is smooth. However, the level surface $\Psi = 0$ and the propagating hypersurface $\Gamma(t)$, may change topology, break, merge and form sharp corners as the function Ψ evolves.

- As $\Psi(x,t)$ remains a function as it evolves, numerical simulations can be developed using a discrete grid in the domain of x and substitution of finite difference approximation for the spatial and temporal derivatives. For example, using a uniform mesh of spacing h, with grid nodes (i,j), and employing the standard notation that Ψ_{ij}^n is the approximation to the solution $\Psi(ih, jh, n\Delta t)$ where Δt is the time step,it can be written as

$$\frac{\Psi_{ij}^{n+1} - \Psi_{ij}^n}{\Delta t} + (F) \cdot |\nabla_{ij} \Psi_{ij}^n| = 0.$$

Here a forward difference scheme in time has been used and $|\nabla_{ij}\Psi^n_{ij}|$ represents some appropriate finite difference operator for the spatial derivatives. Thus an explicit finite difference approach is possible.

- Intrinsic geometric properties of the front are easily determined from the level set function Ψ. For example, at any point of the front, the normal vector is given by,

$$\vec{n} = \frac{\nabla\Psi}{|\nabla\Psi|} \quad .$$

and the curvature of each level set is easily obtained from the divergence of the unit normal vector to the front.

$$\begin{aligned}
k &= \nabla \cdot \frac{\nabla\Psi}{|\nabla\Psi|}. \\
&= (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}) \cdot \frac{(\frac{\partial\Psi}{\partial x}, \frac{\partial\Psi}{\partial y})}{\sqrt{(\frac{\partial\Psi}{\partial x})^2 + (\frac{\partial\Psi}{\partial y})^2}} \\
&= (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}) \cdot \frac{(\Psi_x, \Psi_y)}{(\Psi_x^2 + \Psi_y^2)^{\frac{1}{2}}} \\
&= \frac{\partial}{\partial x} \cdot \frac{\Psi_x}{(\Psi_x^2 + \Psi_y^2)^{\frac{1}{2}}} + \frac{\partial}{\partial y} \cdot \frac{\Psi_y}{(\Psi_x^2 + \Psi_y^2)^{\frac{1}{2}}} \\
&= \frac{\Psi_{xx}\Psi_y^2 - \Psi_x\Psi_y\Psi_{xy}}{(\Psi_x^2 + \Psi_y^2)^{\frac{3}{2}}} + \frac{\Psi_{yy}\Psi_x^2 - \Psi_x\Psi_y\Psi_{xy}}{(\Psi_x^2 + \Psi_y^2)^{\frac{3}{2}}} \\
&= \frac{\Psi_{xx}\Psi_y^2 - 2\Psi_x\Psi_y\Psi_{xy} + \Psi_{yy}\Psi_x^2}{(\Psi_x^2 + \Psi_y^2)^{\frac{3}{2}}}
\end{aligned} \tag{2.5}$$

- There are no significant changes required to follow fronts in three space dimensions. By extending the array structures and gradient operator, propagating surfaces are easily handled.

2.2.2 A Stationary Level Set Formulation

In the Level Set equation

$$\Psi_t + F|\nabla\Psi| = 0$$

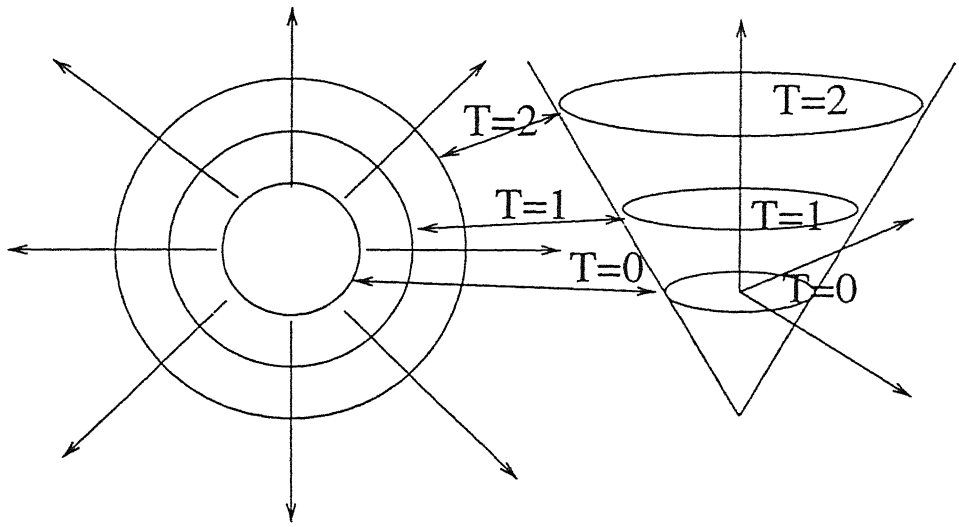


Figure 2.3: Plot of stationary Level Set surface $T(x, y)$

the position of the front is given by the zero level set of Ψ at a time t. Suppose attention is restricted to the particular case of a front propagating with a speed F that is either always positive or negative. In this case, the level set formulation can be converted from a time dependent partial differential equation to a stationary one in which time has disappeared.

The two dimensional case in which the interface is a propagating curve, and suppose we plot the evolving zero level set above the xy plane. That is let $T(x, y)$ be the time at which the curve crosses the point (x, y). The surface $T(x, y)$ then satisfied the equation

$$|\nabla T|F = 1$$

In Figure 2.3 we show a circular front expanding with unit speed, together with the surface $T(x, y)$. From the above equation it is clear that the gradient of arrival time surface is inversely proportional to the speed of the front. This is a Hamilton-Jacobi equation and the recasting of a front motion problem into a stationary one is common in variety of application. If the speed function is always unidirectional i.e. either positive or negative then the crossing time surface $T(x, y)$ is single valued. In

other words both the formulation can be written as follows

- In the time-dependent Level Set equation, the position of the front Γ at time t is given by the zero Level Set of Ψ at time t;that is

$$\Gamma(t) = \{(x, y)|\Psi(x, y, t) = 0\}.$$

- In the stationary Level Set equation, the position of the front Γ is given by the Level Set of value t of the function $T(x, y)$;that is

$$\Gamma(t) = \{(x, y)|T(x, y) = t\}.$$

2.3 Techniques for Tracking Interfaces

2.3.1 Marker/String Methods

A standard approach to modeling moving front comes from discretizing the Lagrangian form of the equations of motion given by equation 2.1. In this technique, the parameterization is discretized into a set of marker particles whose position at any time is used to reconstruct the front. This approach is known under a variety of names, including marker particle techniques, string methods and nodal methods. In two dimensions, the front may be reconstructed as line segments; in three dimensions, triangles can be chosen.

This approach can be illustrated through a straightforward scheme, that constructs a simple difference approximation to the Lagrangian equation of motion. The parameterization interval $[0, S]$ is divided into M equal intervals of size Δs, yielding $M + 1$ mesh points $S_i = i\Delta s$, $i = 0, ...M$. And also the time is divided into equal intervals of length Δt. The image of each mesh point $i\Delta s$ at each time step $n\Delta t$ is a marker point (x_i^n, y_i^n) on the moving front. The goal, is a numerical algorithm that produces new values (x_i^{n+1}, y_i^{n+1}) from the previous position.In the Figure 2.4

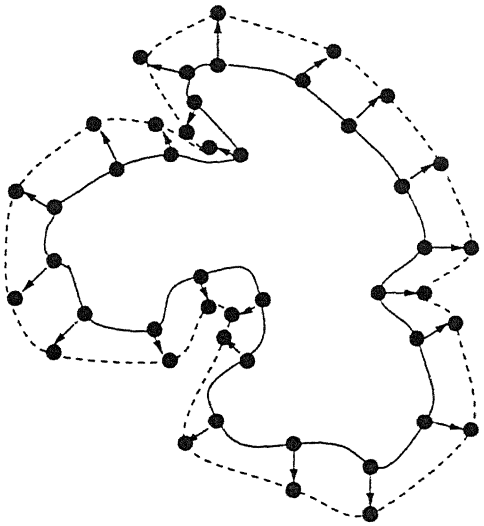


Figure 2.4: Discrete Parameterization of Curve

the outward movement front is shown where the dashed line curve is the curve after some time step. All the black circles in the initial fronts are the marker point. In this figure the movement has been shown in outward normal direction. The parameter derivatives at each marker point is approximated by using neighboring mesh points. Central difference approximation based on Taylor series yield,

$$\begin{aligned} \frac{dx_i^n}{ds} &\approx \frac{x_{i+1}^n - x_{i-1}^n}{2\Delta s} \\ \frac{dy_i^n}{ds} &\approx \frac{y_{i+1}^n - y_{i-1}^n}{2\Delta s} \end{aligned} \tag{2.6}$$

$$\begin{aligned} \frac{d^2x_i^n}{ds^2} &\approx \frac{x_{i+1}^n - 2x_i^n + x_{i-1}^n}{\Delta s^2} \\ \frac{d^2y_i^n}{ds^2} &\approx \frac{y_{i+1}^n - 2y_i^n + y_{i-1}^n}{\Delta s^2} \end{aligned} \tag{2.7}$$

Similarly time derivatives may be replaced by the forward difference approximations.

$$\begin{aligned} \frac{dx_i^n}{dt} &\approx \frac{x_i^{n+1} - x_i^n}{\Delta t} \\ \frac{dy_i^n}{dt} &\approx \frac{y_i^{n+1} - y_i^n}{\Delta t} \end{aligned} \tag{2.8}$$

Substitution of these approximation into the equation of motion of 2.1 gives the

expression for updating the value of the initial curve in the grid.

$$(x_i^{n+1}, y_i^{n+1}) = (x_i^n, y_i^n) + \Delta t\ F(k_i^n)\ \frac{(y_{i+1}^n - y_{i-1}^n) - (x_{i+1}^n - x_{i-1}^n)}{((x_{i+1}^n - x_{i-1}^n)^2 + (y_{i+1}^n - y_{i-1}^n)^2)^{\frac{3}{2}}} \tag{2.9}$$

where

$$k_i^n = 4\ \frac{(y_{i+1}^n - 2y_i^n + y_{i-1}^n)(x_{i+1}^n - x_{i-1}^n) - (x_{i+1}^n - 2x_i^n + x_{i-1}^n)(y_{i+1}^n - y_{i-1}^n)}{((x_{i+1}^n - x_{i-1}^n)^2 - (y_{i+1}^n - y_{i-1}^n)^2)^{\frac{3}{2}}}$$

Using the periodicity of the curve, this is the complete recipe for updating the positions of the particles from one time step to the next.

2.3.2 Volume-of-Fluid Technique

A significantly different approach to front motion is provided by volume of fluid techniques, introduced by Noh and Woodward [7] and is based instead on an Eulerian view. By this method the computation domain is divided into a fixed grid and to each grid cell a value is assigned based on the fraction of the cell containing material inside the interface. For example given a curve, the value of unity is assigned to those cells completely inside this curve, a cell value zero to those completely outside and a fraction between 0 and 1 to cells that straddle the interface, based on the amount of the cell inside the circle.

In order to evolve the interface, the idea is to update the cell fractions on this fixed grid to reflect the progress of the front. Suppose that we wish to advect the front passively under the transport velocity. Noh and Woodward has provided a methodology in which the value in each cell is updated under this transport velocity in each co-ordinate direction by locally reconstructing the front.

In the Figure 2.5 it has been shown how a front propagates in volume fraction method. For simplicity motion, only in vertical direction, is considered. However in this thesis the gradient method has been used which is described below.

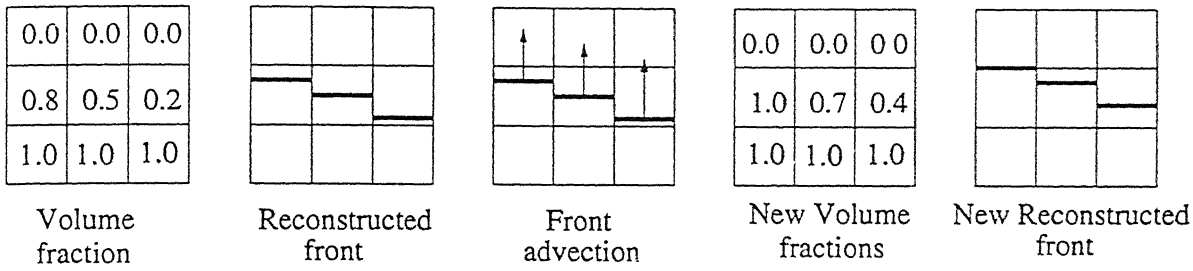


Figure 2.5: Reconstruction and advection of volume fraction

2.3.3 Gradient method

Given $\Psi(x, t = 0)$, the level set equation is written as

$$\Psi_t + F|\nabla\Psi| = 0$$

As discussed earlier the marker particle method discretizes the front. The volume-of-fluid (VOF) method divides the domain space into cells that contain fractions of material. The present (level set) method divides the domain into grid points that hold approximations to the values of the level set functions Ψ. Thus, the grid values give the height of a surface above the domain, and slicing this surface by the xy plane extracts the zero level set corresponding to the front.

Another way is to say that each grid point contains the value of the level set function at that point. Thus there is an entire family of contours, only one of which is the zero level set. Rather than moving each of the contours in a Lagrangian fashion, one stands at each grid point and updates its value to correspond to the motion of the surface, thus producing a new contour value at that grid point. Here also, as was done in the Lagrangian case, one can approximate the solution by replacing all spatial derivatives with central differences and the time derivatives with a forward difference.

2.4 A Detailed Hierarchy of Fast Level Set Methods

The level set method presented so far is a relatively straightforward version that can be easily programmed. However it is neither fast, nor does it make efficient use of computational resources. In this section we will consider more sophisticated version of the basic scheme.

2.4.1 Parallel Algorithm

The straightforward approach presented so far is to solve the partial differential equation for the level set function Ψ in the entire computational domain given an initial value. This is called as the full matrix approach as one is updating all the level sets rather than the zero level set corresponding to the front. The advantage of this approach is that the data structures and operations are extremely clear, and it is a good starting point for building the level set codes.

There are a variety of circumstances in which this approach is desirable. As in case of image processing, sometimes it is required the computation over the entire domain. Since each grid point is updated by a nearest neighbor stencil using only grid points on each side, this technique almost falls under the classification of parallel computation. A parallel version of the level set method was developed in Sethian [8] for the connection machine CM-2 and CM-5. In the CM-2, nodes are arranged in hypercube fashion; in the CM-5, nodes are arranged in a fat-tree. A time-explicit second order space method was used to update the level set equation. Output was controlled by linking the level set evolution to a parallel volume rendering routine, with associated display through access to a parallel frame buffer. As expected, the operation count per time step reduces to $O(1)$, since in most cases the full grid can be placed into physical memory. Thus, most applications of updating propagating

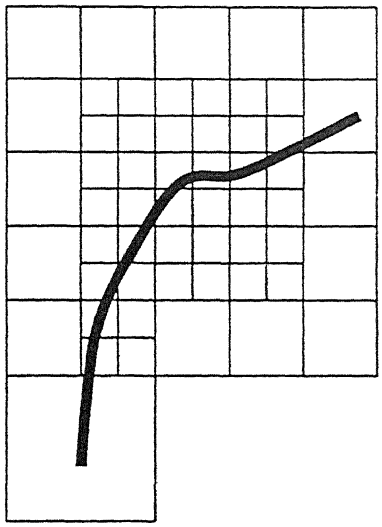


Figure 2.6: Cell hierarchy

interfaces according to given speed functions transpire as real-time movies, the main limitation being the speed of display.

2.4.2 Adaptive Mesh Refinement

One version of an efficient level set method comes from using an adaptive mesh refinement strategy. This is the approach taken by Milne in [9], motivated by the adaptive mesh refinement work by Berger and Colella [10]. Adaptivity may be desired in regions where level curves develop high curvature or where speed function changes rapidly. If the zero level curve identified with a front is the object of interest, then the mesh can be adaptively refined around its location. To illustrate this approach, Figure 2.6 shows mesh cells that are hierarchically refined in response to parent-child relationship around a large curvature in the zero level set Ψ. Calculations are performed on both the fine grids and the coarse grids. The grid cell boundaries always remain parallel to either of the co-ordinate axis and the grids do not over lap. However no attempt was made to align the refined cells with the front.

The data structures for the adaptive mesh refinement are fairly straightforward.

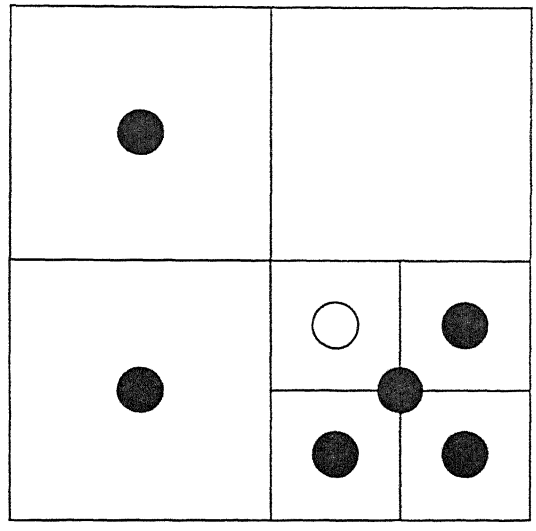


Figure 2.7: Grid values at boundary between refinement levels

However considerable care must be taken at the interfaces between coarse and fine cells. In particular, the update strategy for Ψ at so called "hanging nodes" is subtle. These are nodes at the boundary between two levels of refinement that do not have the full set of nearest neighbors required to update Ψ. To illustrate, Figure 2.7 shows a two dimensional adaptive mesh; the goal is to determine an accurate update strategy for the hanging node marked o.

The strategy laid out by Milne for updating Ψ at such points is as follows. Let us consider the speed function be

$$F(\kappa) = 1 - \epsilon\kappa, \tag{2.10}$$

where ϵ is a constant.

- The advection term (the term independent of κ) of the above equation leads to a hyperbolic equation. Here, straightforward interpolation of the updated values of Ψ from the coarse cell grid is used to produce the new value of Ψ at o.

- In the case of the curvature term $-\epsilon\kappa$, the situation is not so straightforward,

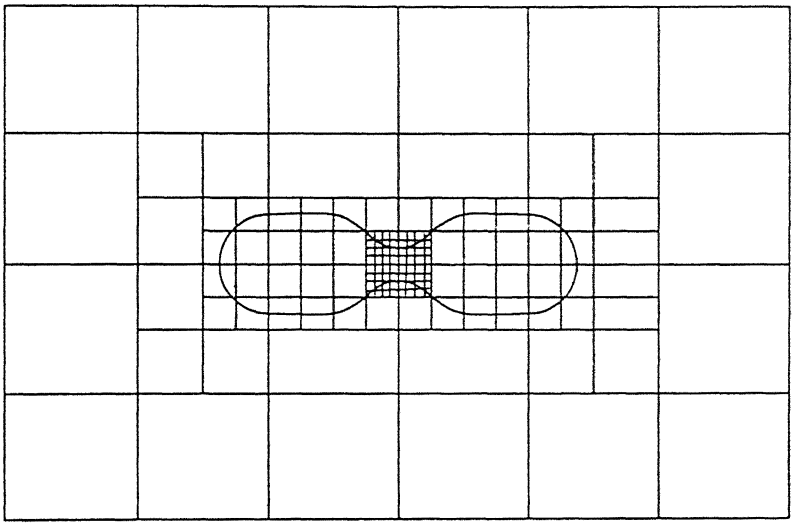


Figure 2.8: Two dimensional slice of adaptive mesh for propagating surface

since this corresponds to a parabolic term that cannot be approximated through simple interpolation. Straightforward interpolation from updated values on the coarse grid to the fine grid provides poor answers. If this procedure is employed, the boundary between the two levels of refinement acts as a source of noise, and significant error is generated at the boundary. In fact, such an approach tested against the simple heat equation using a coarse/fine mesh produces more error in the computed solution than would be produced using a coarse mesh everywhere. Instead, Milne devised the following technique. Values from both the coarse and refined grid around the hanging node are used to construct a least squares solution for Ψ before the update. This solution surface is then formally differentiated to produce the various first and second derivatives in each component direction. These values are then used to produce the updated value for Ψ as done for all other nodes.

As an illustration, in figure 2.8, we show a two-dimensional slice of a fully three-dimensional adaptive mesh calculation of a surface collapsing under its mean curvature.

2.4.3 Narrow Banding and Fast Methods

There are several disadvantages with the full matrix approach given section 2.4.1 if one is only interested in a specific front. These disadvantages are mainly manifested in the speed of computation and calculation of the front speed. The narrow band technique helps to remove them as explained below.

- *Speed* : Performing calculations over the entire computational domain requires $O(N^m)$ operations, where N is the number of grid points along a side and m is the dimension of computational domain. As an alternative, an efficient modification is to perform operation only in a neighborhood of the zero level set. This is known as the *narrow band approach*. In this case, the operation count in three dimensions drops down to $O(kN^2)$, where k is the number of cells in the narrow band. Thus, a significant cost reduction is achieved.

- *Calculating Extension Variables* : The level set approach requires the extension of the speed function F in equation (2.4) to all of space. This speed function is then updated not only in the zero level set but also for others. As described earlier, three types of arguments may influence the front speed F; local, global, and independent. Some of these variables may have meaning only on the front itself, and it may be both difficult and awkward to design a speed function that extrapolates the velocity away from the zero level set in a smooth fashion. Thus, another advantage of the narrow band approach is that this extension need only be done to points lying in the narrow band, as opposed to all points in the computational domain.

The above narrow band method was introduced in Chopp [11]. Figure 2.9 shows the placement of a narrow band around the familiar initial front. The entire two-dimensional grid of data is stored in a square array(dark grid points in figure 2.9 are located in a narrow band around the front of a user defined width)(see Figure 2.10).

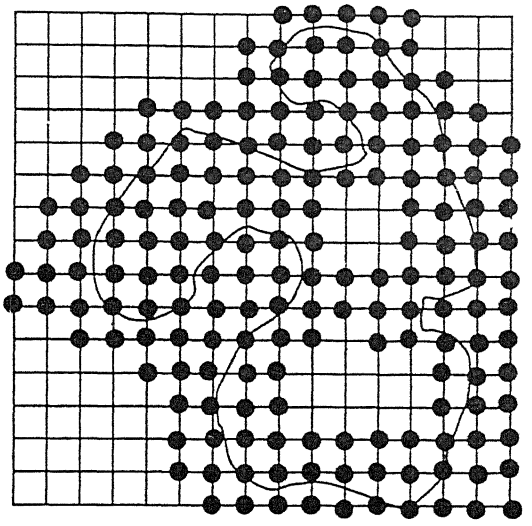


Figure 2.9: Dark grid points are members of narrow band

Only the values of Ψ at such points within the tube are updated.Values of Ψ at grid points on the boundary of the narrow band are frozen. When the front moves near the edge of the tube boundary, the calculation is stopped, and a new tube is built with the zero level set interface boundary at the center. This rebuilding process is known as re-initialization.

Thus, the narrow band method consists of the following loop.

- Tag alive points in narrow band.
- Build land mines to indicate near edge.
- Initialize far away points outside(inside) narrow band with large positive (negative) values.
- Solve level set equation until land mine hit.
- Rebuild, loop.

Use of narrow bands leads to level set front advancement algorithms that are computationally equivalent in terms of complexity to traditional marker methods and cell

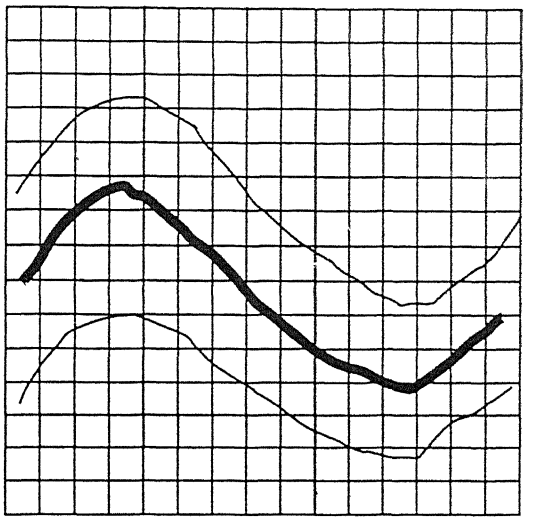


Figure 2.10: An zero level set with the narrow band boundary

techniques. This have the advantages of handling topological changes and are more accurate and easily extendable to multi dimension. Typically, the speed associated with the narrow band method is about ten times faster on a 160x160 grid than the full matrix method. Such a speed-up is substantial, particularly in three-dimensional simulations. It can make the difference between computationally intensive problems and those that can be done with relative ease. This narrow banding technique requires rebuilding and re-initializing a new narrow band around the location of the front. Some of the ways are described in the following section.

2.4.4 Re-Initialization Techniques

2.4.4.1 Direct Evaluation

A straightforward re-initialization technique to rebuild the band is to first find the zero level set by using a contour plotter and then recalculate the signed-distance from each grid point to this zero level set. This technique can be used to ensure that the level set function stays well behaved. However it has got some drawbacks itself.

First, one must find the front itself. For two dimensions, one can use a contour plotter and in three dimensions, some version of Lorenson and Cline's [12] voxel scheme is possible. Nonetheless, any level set scheme tries to avoid finding the front. since it introduces considerable complication to the technique. Second, such an approach is expensive. For example, in two-dimension a representation of the front as N segments will require N evaluation to find the distance to each of N^2 grid points. This is an $O(N^3)$ calculations which is N times more expensive than updating the level set function Ψ over the entire grid.

2.4.4.2 Iteration

An alternative to this was given by Sussman et al. [13]. Its virtue is that one need not find the zero level set to re-initialize the level set function. Let us consider the partial differential equation

$$\Psi_t = sign(\Psi)(1 - |\nabla\Psi|) \tag{2.11}$$

where $sign(\Psi)$ gives the sign of Ψ. Given any initial data for Ψ, the steady state solution of the above equation provides a new value of Ψ for which $|\nabla\Psi| = 1$. The sign function controls the direction of the information flow. The net effect is to straighten out the level sets on either side of the zero level set and produce a Ψ function with $|\nabla\Psi| = 1$ corresponding to the signed distance function. Thus, their approach is to stop the level set calculation periodically and solve the above until convergence. If done often enough, the initial guess is often close to the signed-distance function and few iterations are required. One potential disadvantage of the above scheme is the relative crudeness of the switch function based on checking the sign of the level set equation; considerable motion of the zero level set can occur during the re-initialization, since the sign function does not do an accurate job of using information about the exact location of the front. A hybrid method combining

a volume-of-fluid approach and a level set method foe this problem may be found in [14].

2.4.4.3 Dynamic Allocation of Points in Narrow Band

Another approach is to dynamically add grid points to the narrow band as it moves. Thus, points whose Ψ values dip below a certain negative level are removed, while neighbors are added around those that dip below a certain positive value. When new grid points are added, they must be given appropriate Ψ values. This is accomplished by re-initializing every time step, usually by the above iterative technique, to return to the signed-distance function. Thus, new grid points are added, the Ψ function in the entire narrow band is re-initialized, and the calculation is advanced one time step.

2.4.4.4 Huyghen's Principle

An alternative technique, described in Sethian [15], is based on the idea of computing crossing times as discussed in [16], and is related to the ideas given by Kimmel and Bruckstein [17]. Consider an initial value for the $\Psi(x,t)$. The goal is to produce a new level set function $\Psi(x,t)$ with the zero level set unchanged and that corresponds to the signed-distance function around that zero level set. This new function may be built as follows. With speed function $F = 1$, flow the level set function both forwards and backwards in time and calculate crossing times(that is when Ψ changes sign) at each grid point. These crossing times (both positive and negative) are equal to the signed-distance function by Huyghen's principle. This approach has the advantage that one knows how long one must run the problem forward and backward to re-initialize grid points at a given distance from the front, since one is using a speed function of unity. One can perform this iteration using a high order scheme to produce accurate values for the crossing times.

This idea of "computing crossing time" is equivalent to converting the level set

evolution problem into the stationary problem that was discussed earlier. In this converted state, we can develop an ultra-fast level set scheme for the particular case of solving the level set equation for speed function $F = F(x, y, z)$, where F is always either positive or negative.

Chapter 3

Shape Recovery With Front Propagation

In this chapter we will discuss about the various steps we have followed for the shape recovery in medical imaging.

3.1 Level Set Approach

The initial snake, which is a closed curve without overlaps, is placed entirely inside a given shape that is to be recovered. We call this curve the initial front. Now we let the front grow with a speed depending on its curvature and the image gradient. The image gradient can be seen as landscape, where regions with relatively low gradient are represented by high planes or mountains and regions with relatively high gradient by valleys (a very high gradient can then be imagined as canyon). Now one can imagine the front as a number of heavy balls connected by elastic strings, placed around the top of the mountain, which represents the center of the object to be recovered. The balls roll down the mountains until they finally come to a halt in deep down of the valleys and canyons. The elasticity of the string is responsible of finding the real canyon (the desired shape), because it does not allow a single ball to rest in

a tiny hole. The other balls will pull out the trapped one because of the elasticity of the connection.

As discussed in the previous chapter the equation for the evolution of zero level set is

$$\Psi_t + F|\nabla\Psi| = 0 \tag{3.1}$$

which can be written as

$$\Psi_t + \sum_{i=1}^{N} \Psi_{x_i x_{i_t}} = 0$$

where x_i is the ith component of x.

Let us discretize the space into a number of grid points with equal spacing, h, among each other. Further the value of Ψ at any such grid point (i, j, k) is denoted by Ψ_{ijk}. Unless otherwise mentioned from now onwards we will consider everything in 3D. Employing the standard notation Ψ^n_{ijk} is the approximation to the solution $\Psi(ih, jh, kh, n\Delta t)$, where Δt is the time step. Then we can write the above equation as

$$\frac{\Psi^{n+1}_{ijk} - \Psi^n_{ijk}}{\Delta t} + F|\nabla_{ijk}\Psi^n_{ijk}| = 0 \tag{3.2}$$

The speed term F is dependent on the curvature κ. We separate $F(\kappa)$ into a constant advection term F_0 and the remainder $F_1(\kappa)$, that is

$$F(\kappa) = F_0 + F_1(\kappa) \tag{3.3}$$

The advection term F_0 defines a uniform speed of the front which would be achieved at regions of the front with zero curvature. The diffusion term $F_1(\kappa)$ smoothes out the high curvature regions and has the same regularizing effect on the front as the internal deformation energy terms in splines. In practice we use the following formula:

$$F = 1 - \epsilon\kappa$$

where ϵ is the entropy condition which regulates the smoothness of the curve.

The curvature is obtained from the divergence of the gradient of the unit normal vector to front, that is

$$\begin{aligned}
\kappa &= \nabla \cdot \frac{\nabla\Psi}{|\nabla\Psi|} \\
&= (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}) \cdot \frac{(\frac{\partial\Psi}{\partial x}, \frac{\partial\Psi}{\partial y}, \frac{\partial\Psi}{\partial z})}{\sqrt{(\frac{\partial\Psi}{\partial x})^2 + (\frac{\partial\Psi}{\partial y})^2 + (\frac{\partial\Psi}{\partial z})^2}} \\
&= (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}) \cdot \frac{(\Psi_x, \Psi_y, \Psi_z)}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{\frac{1}{2}}} \\
&= \frac{\partial}{\partial x} \cdot \frac{\Psi_x}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{\frac{1}{2}}} + \frac{\partial}{\partial y} \cdot \frac{\Psi_y}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{\frac{1}{2}}} + \frac{\partial}{\partial z} \cdot \frac{\Psi_z}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{\frac{1}{2}}} \\
&= \frac{\Psi_{xx}(\Psi_y^2 + \Psi_z^2) + \Psi_{yy}(\Psi_x^2 + \Psi_z^2) + \Psi_{zz}(\Psi_x^2 + \Psi_y^2) - 2\Psi_x\Psi_{xy}\Psi_y - 2\Psi_y\Psi_{yz}\Psi_z - 2\Psi_x\Psi_{xz}\Psi_x}{(\Psi_x^2 + \Psi_y^2 + \Psi_z^2)^{\frac{3}{2}}}
\end{aligned} \tag{3.4}$$

3.1.1 Image Based Speed Term

Till now we have not used any image information. A rather broad discussion of different techniques of integrating image information into the speed term of equation 3.3 is presented in [19]. The problem lies in the fact that the front (and so the zero level set) should be forced to stop in the vicinity of the desired objects boundaries. To achieve this, we have to extend the image based speed term, which is only defined at the zero level set, to the other level sets. Either way, we need to manipulate the generic speed term in equation 3.3 to take the image gradient into account. This is done by multiplying the speed function F with a quantity Φ. The term Φ can be defined many ways. One of the ways proposed by Sethian et. al is

$$\Phi(x, y, z) = \frac{1}{1 + |\nabla G_\sigma * I(x, y, z)|}. \tag{3.5}$$

Where $G_\sigma * I$ denotes the image convolved with a Gaussian smoothing filter whose characteristic width is σ. But this term is proved to fall too slowly to zero in regions

of high image gradient. Instead we used the following speed term which is much faster compared to the above one.

$$\Phi(x,y,z) = e^{-|\nabla G_\sigma * I(x,y,z)|}. \tag{3.6}$$

3.1.2 Discrete Numerical Approximation

As we work in digital images so we need to discretize our formula so that it can be easily implemented in the algorithm.

$$\Psi_x = \Psi_i = \frac{\Psi_{i+1,j,k} - \Psi_{i-1,j,k}}{2\Delta s} \tag{3.7}$$

$$\Psi_y = \Psi_j = \frac{\Psi_{i,j+1,k} - \Psi_{i,j-1,k}}{2\Delta s} \tag{3.8}$$

$$\Psi_z = \Psi_k = \frac{\Psi_{i,j,k+1} - \Psi_{i,j,k-1}}{2\Delta s} \tag{3.9}$$

$$\Psi_{xx} = \Psi_{ii} = \frac{\Psi_{i+1,j,k} - 2\Psi_{i,j,k} + \Psi_{i-1,j,k}}{\Delta s^2} \tag{3.10}$$

$$\Psi_{yy} = \Psi_{jj} = \frac{\Psi_{i,j+1,k} - 2\Psi_{i,j,k} + \Psi_{i,j-1,k}}{\Delta s^2} \tag{3.11}$$

$$\Psi_{zz} = \Psi_{kk} = \frac{\Psi_{i,j,k+1} - 2\Psi_{i,j,k} + \Psi_{i,j,k-1}}{\Delta s^2} \tag{3.12}$$

$$\Psi_{xy} = \Psi_{ij} = \frac{\Psi_{i+1,j+1,k} - \Psi_{i-1,j+1,k} - \Psi_{i+1,j-1,k} + \Psi_{i-1,j-1,k}}{\Delta s^2} \tag{3.13}$$

$$\Psi_{yz} = \Psi_{jk} = \frac{\Psi_{i,j+1,k+1} - \Psi_{i,j+1,k-1} - \Psi_{i,j-1,k+1} + \Psi_{i,j-1,k-1}}{\Delta s^2} \tag{3.14}$$

$$\Psi_{xz} = \Psi_{ik} = \frac{\Psi_{i+1,j,k+1} - \Psi_{i-1,j,k+1} - \Psi_{i+1,j,k-1} + \Psi_{i-1,j,k-1}}{\Delta s^2} \tag{3.15}$$

The norm of the gradient will be

$$|\nabla\Psi| = \sqrt{\Psi_x^2 + \Psi_y^2 + \Psi_z^2} \tag{3.16}$$

These above approximations of different partial differentiation can be applied directly in the equation 3.4 to get the curvature of a point(or pixel) in a digital image. Now the speed term becomes

$$\begin{aligned} F &= \Phi * (1.0 - \epsilon\kappa) \\ &= e^{-|\nabla G_\sigma * I(x,y,z)|} * (1.0 - \epsilon\kappa). \end{aligned} \tag{3.17}$$

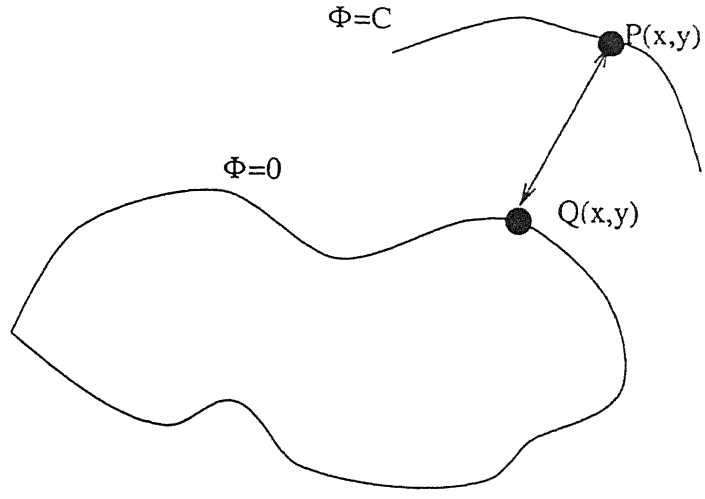


Figure 3.1: Extension of Image Based speed Term

The time derivative for Ψ can be approximated as

$$\Psi_t = \frac{\Psi^{n+1}_{i,j,k} - \Psi^{n}_{i,j,k}}{\Delta t} \tag{3.18}$$

So now putting it in equation 3.1

$$\frac{\Psi^{n+1}_{i,j,k} - \Psi^{n}_{i,j,k}}{\Delta t} + F * |\nabla \Psi| = 0$$

$$\Psi^{n+1}_{i,j,k} = \Psi^{n}_{i,j,k} - \Delta t * F * |\nabla \Psi| \tag{3.19}$$

Finally the equation of Level Set evolution takes the form

$$\Psi^{n+1}_{i,j,k} = \Psi^{n}_{i,j,k} - \Delta t * (e^{-|\nabla G_\sigma * I(x,y,z)|} * (1.0 - \epsilon\kappa)) * |\nabla \Psi| \tag{3.20}$$

3.1.3 Extending The Speed Function

The image-based speed terms have meaning only on the boundary $\Gamma(t)$, that is, on the level set $\Psi = 0$. This follows from the fact that they are designed to force the propagating level set $\Psi = 0$ to a complete stop in the neighborhood of an object boundary. However as discussed earlier also the level set equation of motion is written

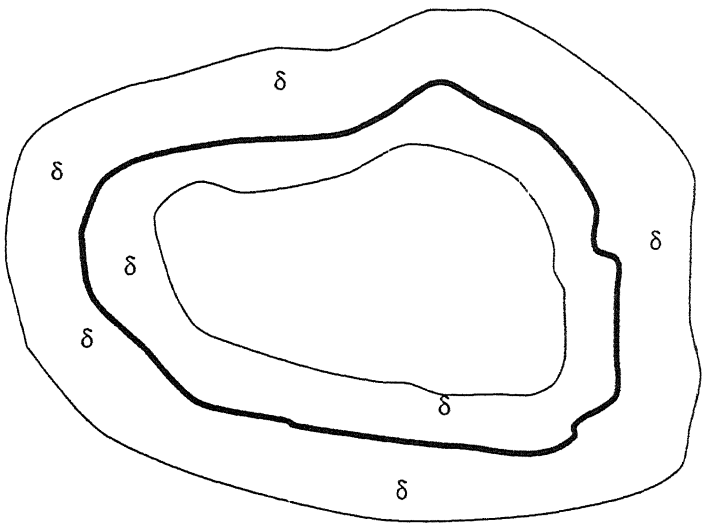


Figure 3.2: Extension of Image Based speed Term

for the function Ψ defined over the entire domain. Consequently, we require that the evolution equation has a consistent physical meaning for all the level sets, that is at every point in the image plane. This technique is well described in Sethian [18]. They have suggested a new speed term and method to extend the speed function globally. As shown in Fig 3.1 we can construct one such extension to the image-based speed function by letting the value of the speed term at a point lying on a level set $\Psi = C$ be the value of the speed term at point Q, such that point Q is closest to P and lies on the level set $\Psi = 0$. By this attempt the collision of level set can be avoided.

3.1.4 Narrow-Band Extension and Re-Initialization

In the last chapter we have described the theory of the narrow-banding and initialization. Here we will review it briefly and will discuss the details that we have implemented. The front can be moved by updating the level set function at a small set of points in the neighborhood of the zero instead of updating it at all the points on the grid. In Figure 3.2 the bold curve depicts the level set $\Psi = 0$ and the region around it is the narrow band. The narrow band is bounded on either side by

two curves which are at a distance δ apart ,that is, the two curves are the level sets $\Psi = \pm\delta/2$. The value of δ determines the number of grid points that fall within the narrow band. Since, during a given time step the value of Ψ_{ijk} is not updated at points lying outside the narrow band, the level sets $|\Psi| > \delta/2$ remain stationary. The zero set which lies inside moves until it collides with the boundary of the narrow band. Which boundary the front collides depends on whether it is moving inward or outward. As the consequence of the update strategy, the front can be moved through a maximum distance of $\delta/2$, either inward or outward , at which point we must build a new appropriate narrow band. Here we have reinitialized the Ψ function by treating the current zero level set as the initial curve $\Gamma(0)$. Once a new Ψ function is defined on the grid, we can create a narrow band around the zero set, and go through another set of, say, l iterations in time to move the front ahead by a distance equal to $\delta/2$. The value of l is set to the number of time steps required to move the front by a distance roughly equal to $\delta/2$. We have taken the inner loop l as 50 or 40. In both the case it works fine.

3.1.5 Straightforward Narrow-Band Extension

The narrow-band approach, in addition to being computationally efficient, allows us to return to the original construction of the speed function extension and replace it with a more mathematically appealing vision. Since the narrow-band mechanism periodically recalibrates the front, we can in fact simply move each level set with the speed determined by the image gradient as given in the equation 3.5 and 3.6. In other words, for points inside the narrow band, the external speed values are picked directly from their corresponding image locations. Thus we can ignore the previous extension velocity suggested in Sethian [18] and provide a purely geometric one based on the local image gradient. Although this may cause many other level sets to temporarily stop, the narrow-band re-initialization resets them all around the zero level set. This

will ensure that the zero level set is drawn close to the object boundary as well as retain other desirable properties of the level set approach, such as topological merge and split. Also, since the extension computation does not involve any search (as in earlier case the searched nearest point on the zero level set from a point gets the speed term same as that of the searched point), the time complexity of this approach is identical to that of a basic narrow-band front propagation algorithm. We have used this computationally efficient algorithm.

3.1.6 Algorithm

Following the algorithm that we have followed for both 2D and 3D segmentation. Algorithm:

1. Read the image data i.e. intensity value of each pixel and stored as $I(x, y, z)$.
2. Convolve the image with a Gaussian smoothing filter.
3. Calculate the gradient of the image.
4. Initialize the value of Ψ calculating the distance from the initial front and calculate the narrow band corresponding to this initial front.
5. Set the iteration number $m = 0$ and go to step 6
6. At each grid-point (i, j, k) lying inside the narrow-band, compute the extension of image based speed term.
7. With the above value of the speed term and $\Psi^n_{i,j,k}$, calculate $\Psi^{n+1}_{i,j,k}$ using the upwind, finite difference scheme as described in earlier equations.
8. Construct a polygonal approximation for the level set $\Psi = 0$ from $\Psi^{m+1}_{i,j,k}$. A contour tracing procedure is used to obtain a polygonal approximation. Given

a cell (i, j, k) which contains $\Gamma(t)$, this procedure traces the contour by scanning the neighboring cells in order to find the next cell which contains $\Gamma(t)$. Once such a cell is found the process is repeated until the contour closes on itself. The set of nodes visited during this tracing process constitute the polygonal approximation to $\Gamma(t)$. In general, to collect all the closed contours, the above tracing procedure is started at a new, as yet unvisited, cell which contains the level set $\Psi = 0$. A polygonal approximation is required in step 6 for the evaluation of image-based speed term and more importantly, in step 10 for reinitializing the Ψ function.

9. Increment m by one. If the value of m equals l, go to step 10, else, go to step 6.

10. Compute the value of signed distance function Ψ by treating the polygonal approximation of $\Psi = 0$ as the initial contour $\Gamma(0)$. As mentioned earlier, a more general method of re-initialization is required when $\Psi = 0$ changes topology. Go to step 1.

Chapter 4

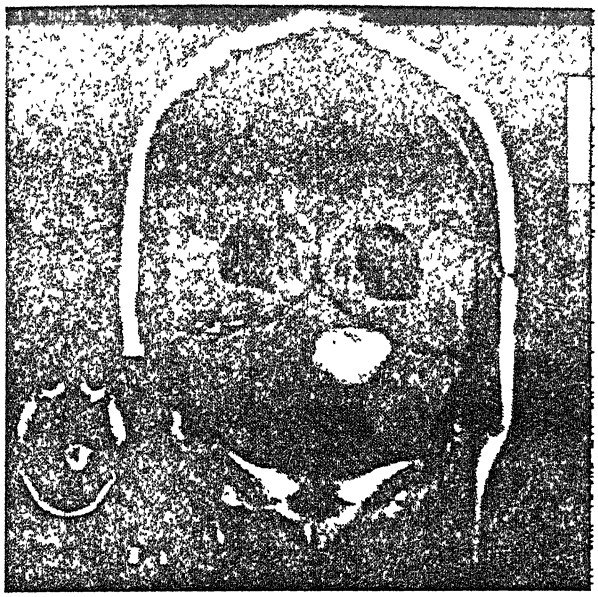
Results and Conclusion

In this chapter we will present all our results with a pertinent discussion and we will conclude the thesis suggesting some future work.

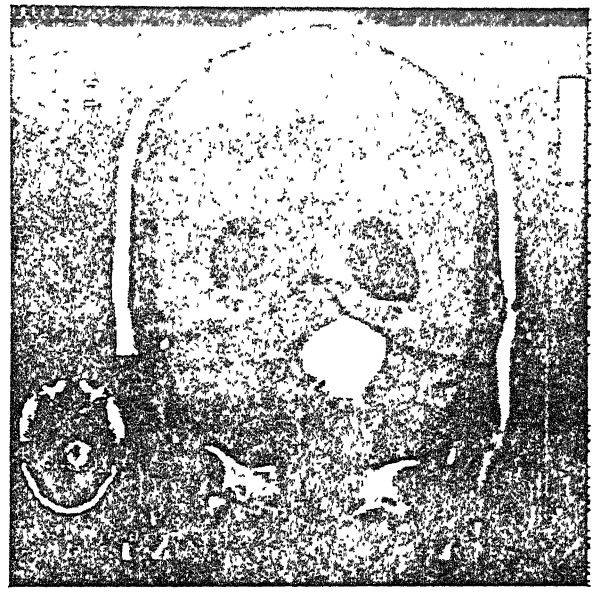
4.1 Results

We have used MRI (Magnetic Resonance Imaging) images for the detection of boundaries. We have used image of dimensions of 256x256 square image with a 8-bit grey scale value. Our methods require an initial curve (which will be treated as zero level set) completely inside the boundary of the objects whose shape is to be determined. However, the initial contour can be placed anywhere in the image plane. Our front seeks the object boundaries by either propagating inward or outward in the normal direction. This choice is made at the time of initialization. We have chosen to give control of certain parameters used, to the user at the time of initialization. In 2D the user can specify an initial curve by specifying the center co-ordinates (x_c, y_c) and the radius r of a circle. The narrow-band width δ can be given an integer value. The time step dt is also user specified. Though parameters like dt, δ can be hard coded in the program. But we found it more suitable to give the control in user's hand. Results are shown in the following pages.

Input Slices

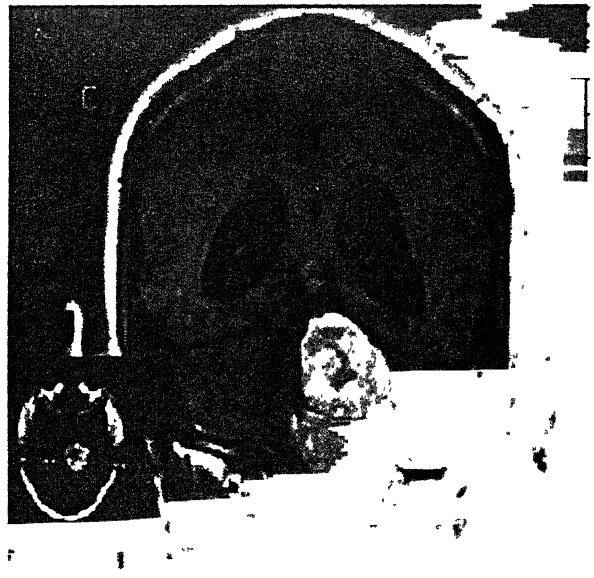


(a) slice 1



(b) slice 2

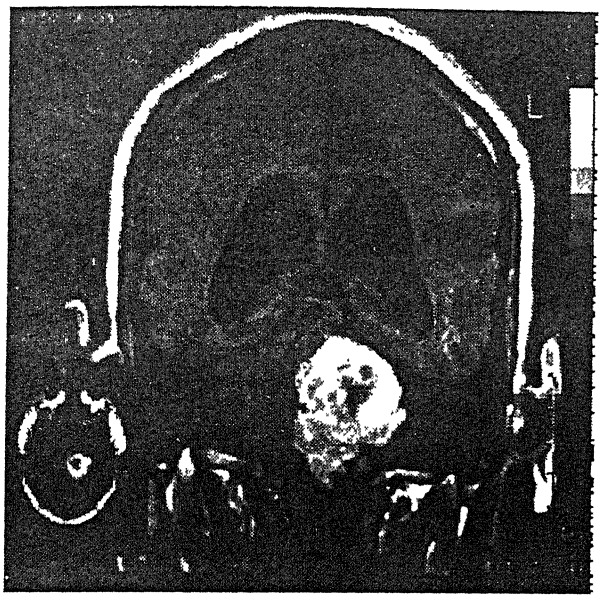
(c) slice 3



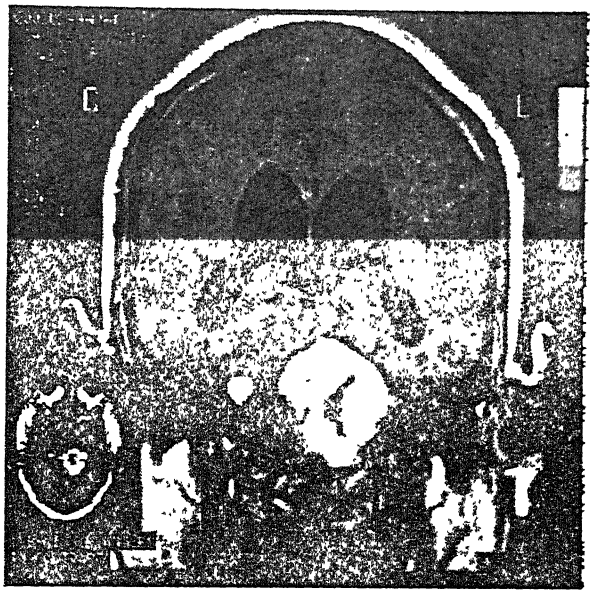
(d) slice 4

Figure 4.1: MRI slices of the brain used as input image.

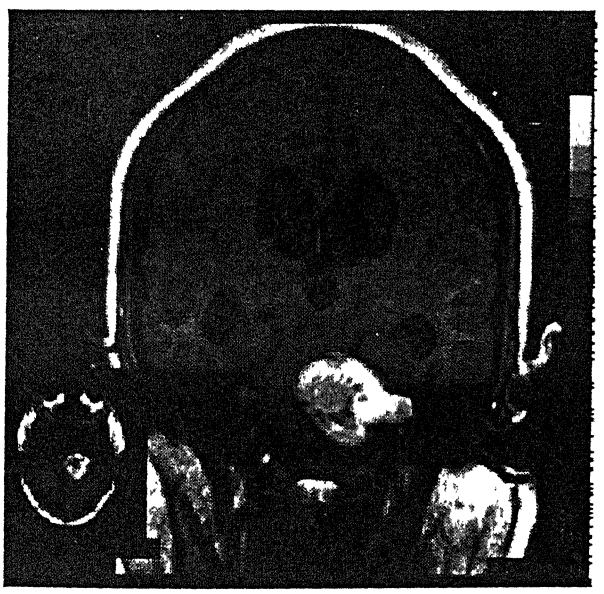
Input Slices



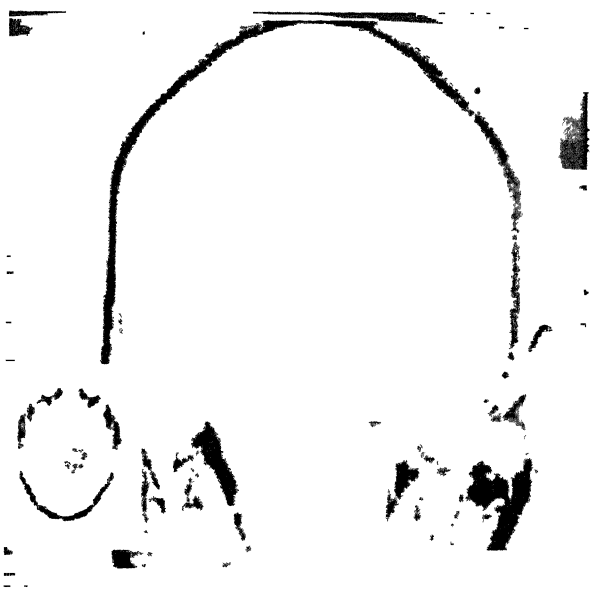
(e) slice 5



(f) slice 6



(g) slice 7



(h) slice 8

Figure 4.1: MRI slices of the brain used as input image. (*contd.*)

2D Results

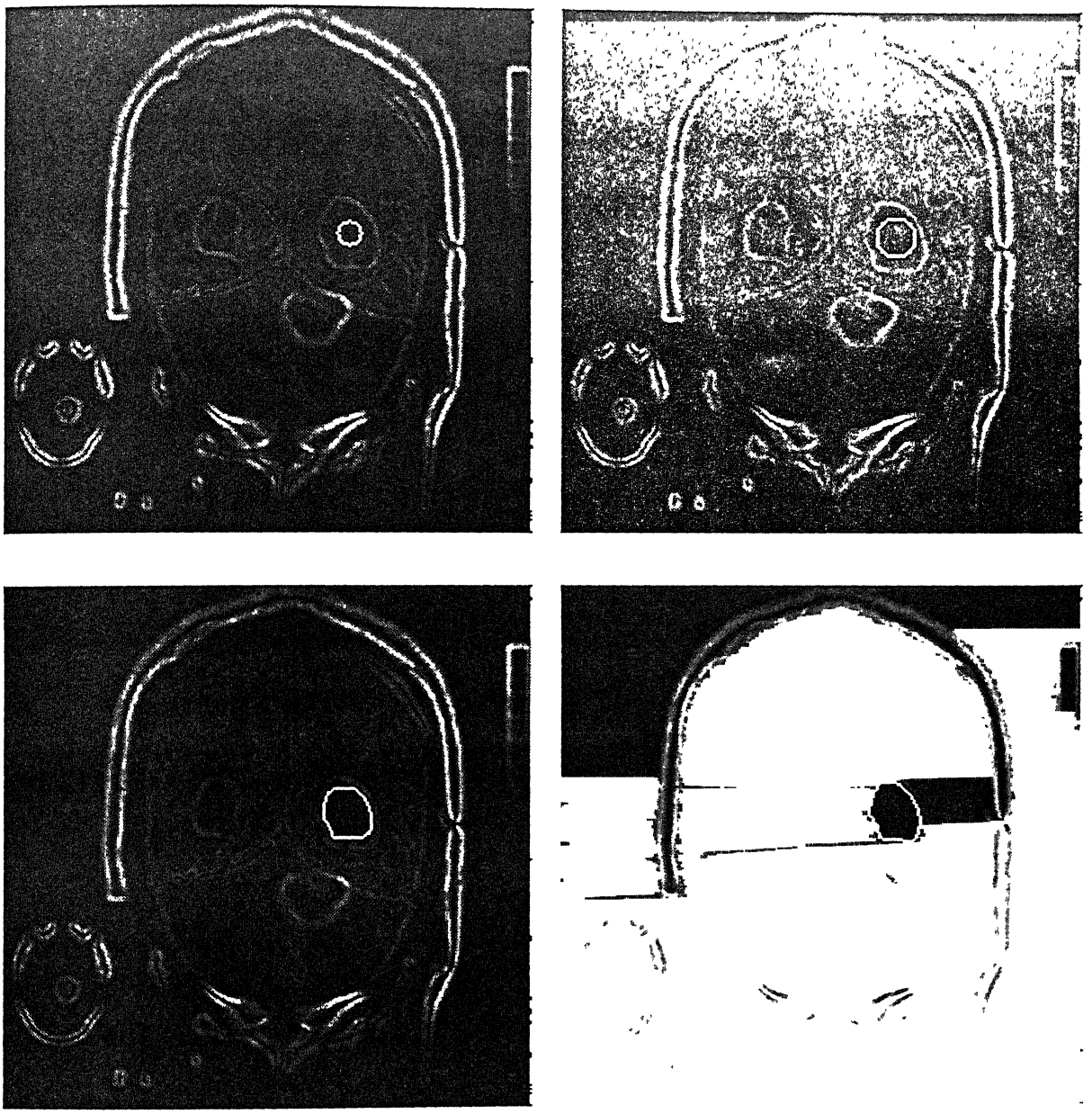


Figure 4.2: Finished in 3 steps with the parameters $dt = 0.12,\ \epsilon = 0.5$

In this page the top-left figure shows the initial curve(circle of radius 6 centered at $x_c = 111, y_c = 165$)) specified by the user. Top-right figure shows the front after 1 step and figure bottom-left and bottom-right follows as step 2 and step 3 respectively.

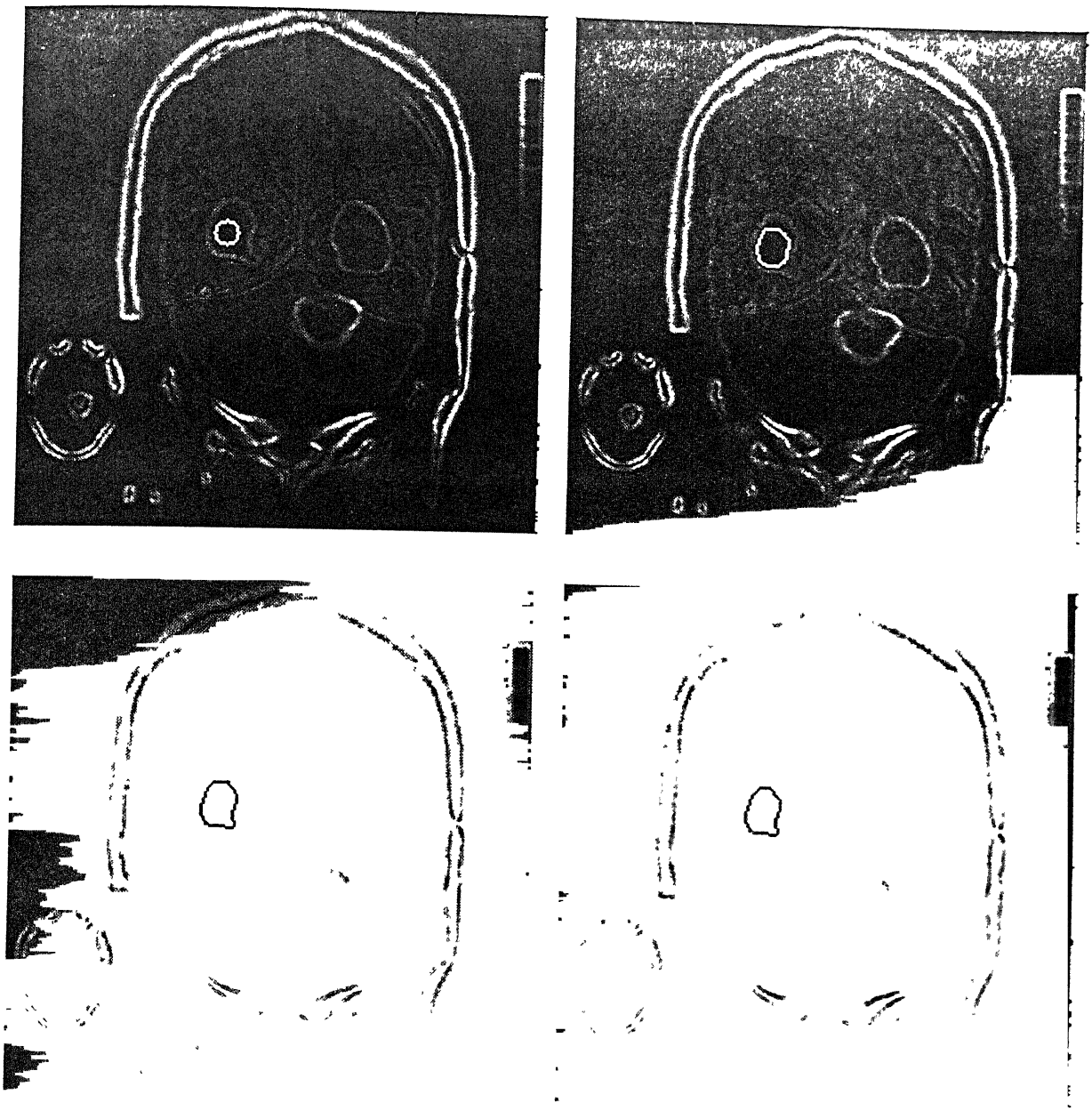


Figure 4.3: Finished in 3 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8.$

Here also an initial circle of radius 6 is put in the top-left figure with the center at $x_c = 110, y_c = 164$. The required boundary is being detected in 3 steps. So the top-right, bottom-left and bottom-right figure follows at step 1, step 2, step 3, respectively. The parameters are same as in that of the previous page.

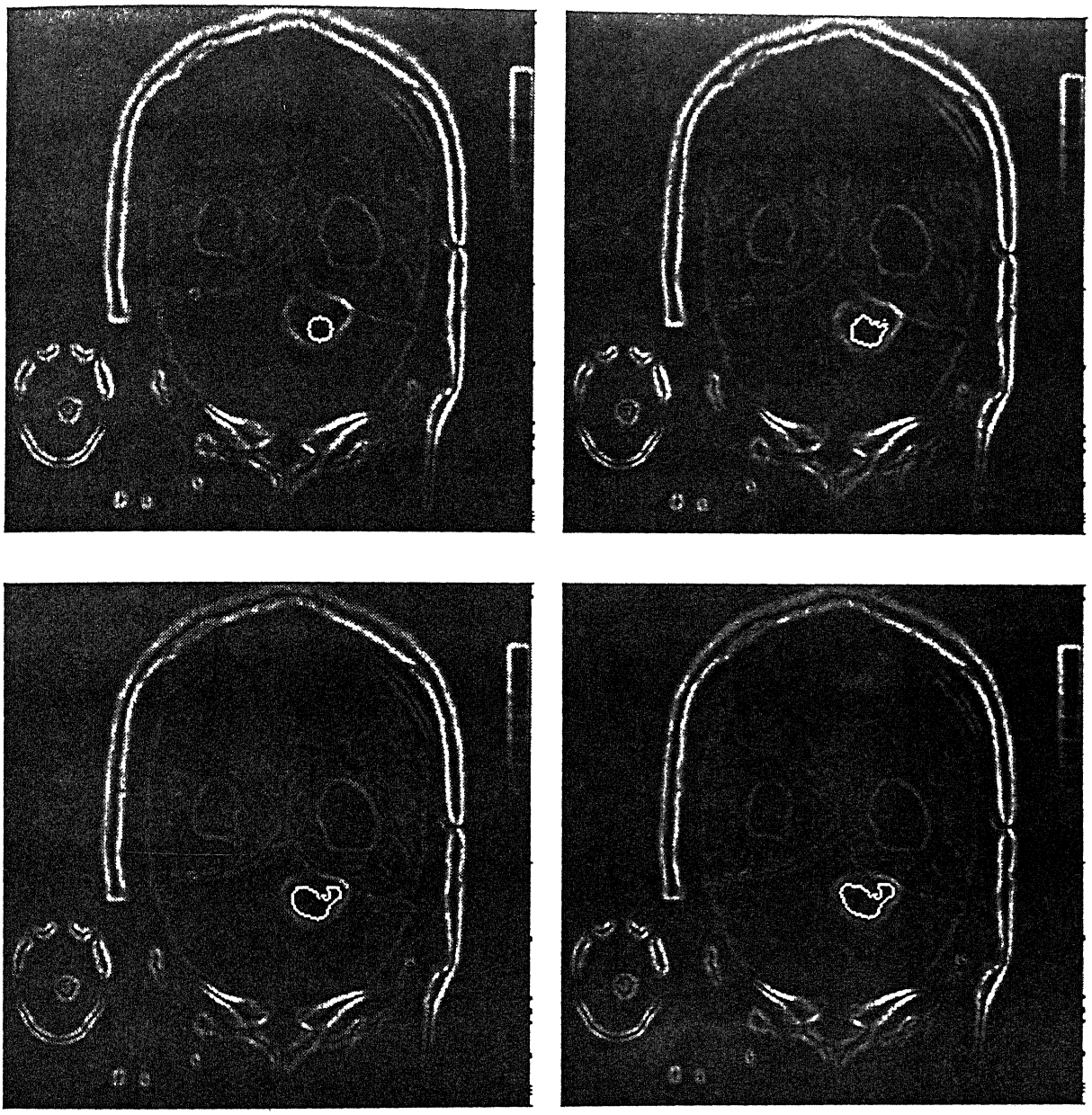
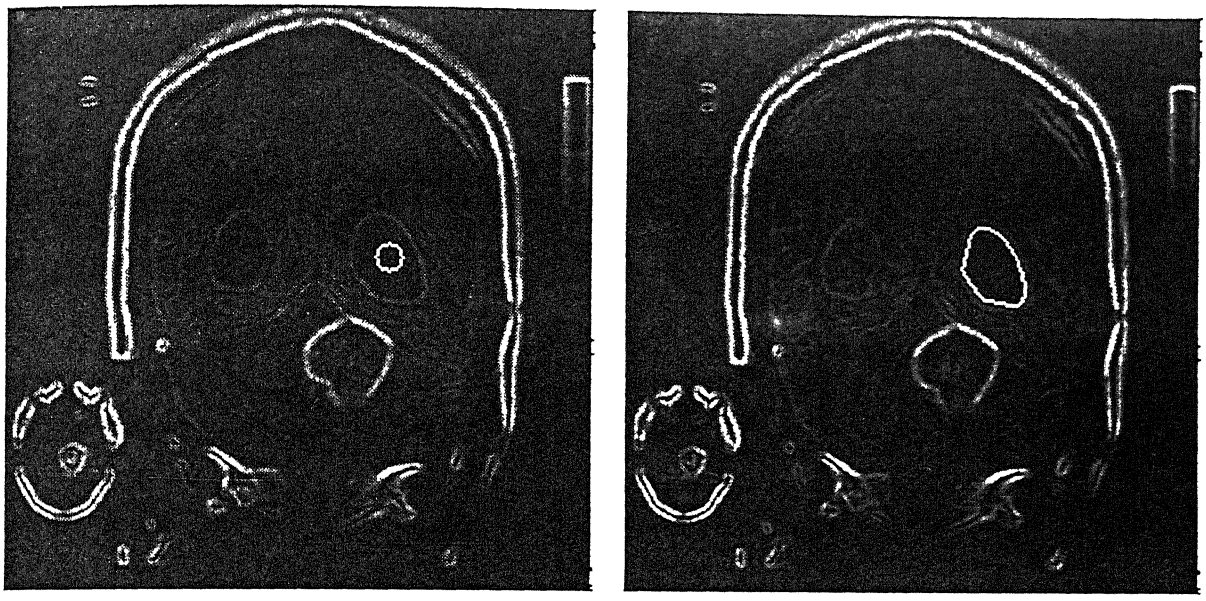


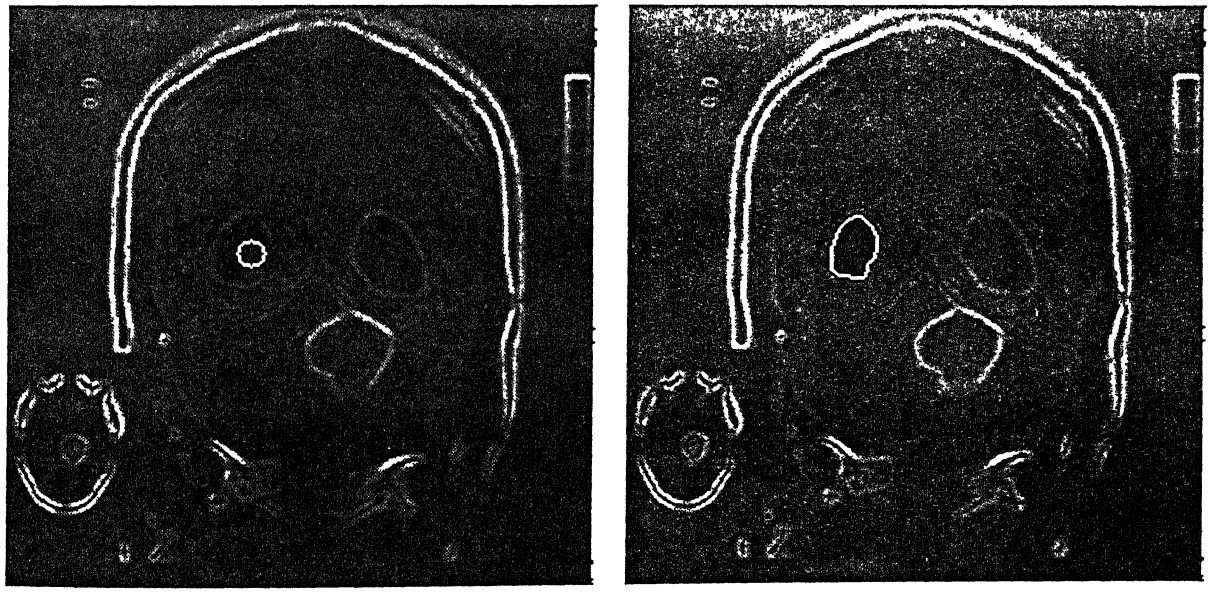
Figure 4.4: Finished in 3 steps with the parameters $dt = 0.4,\ \epsilon = 0.01,\ \delta = 8$.

Here also the initial circle of radius 6 is placed at $x_c = 150, y_c = 156$. And step 1, step 2, step 3 follows as the top-right, bottom-left, bottom-right figure. But here the parameters are different. Here the time step dt is more and ϵ is less as the boundary has many high curvature which needs more smoothing that is being achieved by increasing ϵ and this time step is found to be suitable for this case experimentally.



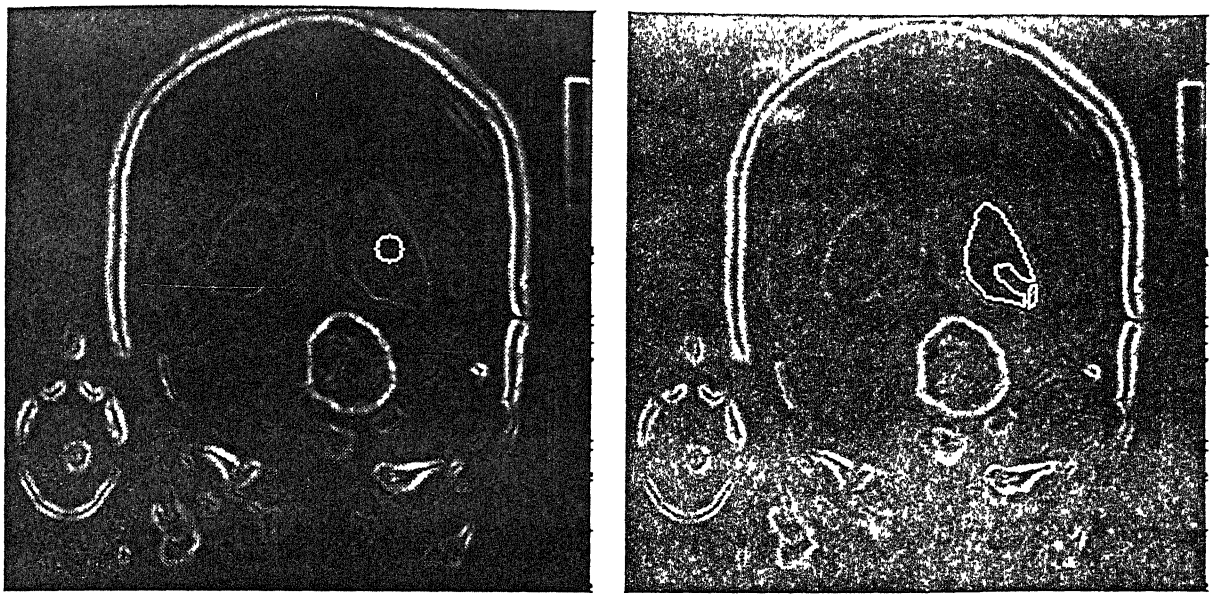
(a) Initial Curve($x_c = 110,\ y_c = 164,\ r = 6$) (b) at step 7

Figure 4.5: Result after 7 steps with the parameters $dt = 0.12,\ \epsilon = 0\ 5,\ \delta = 8.$



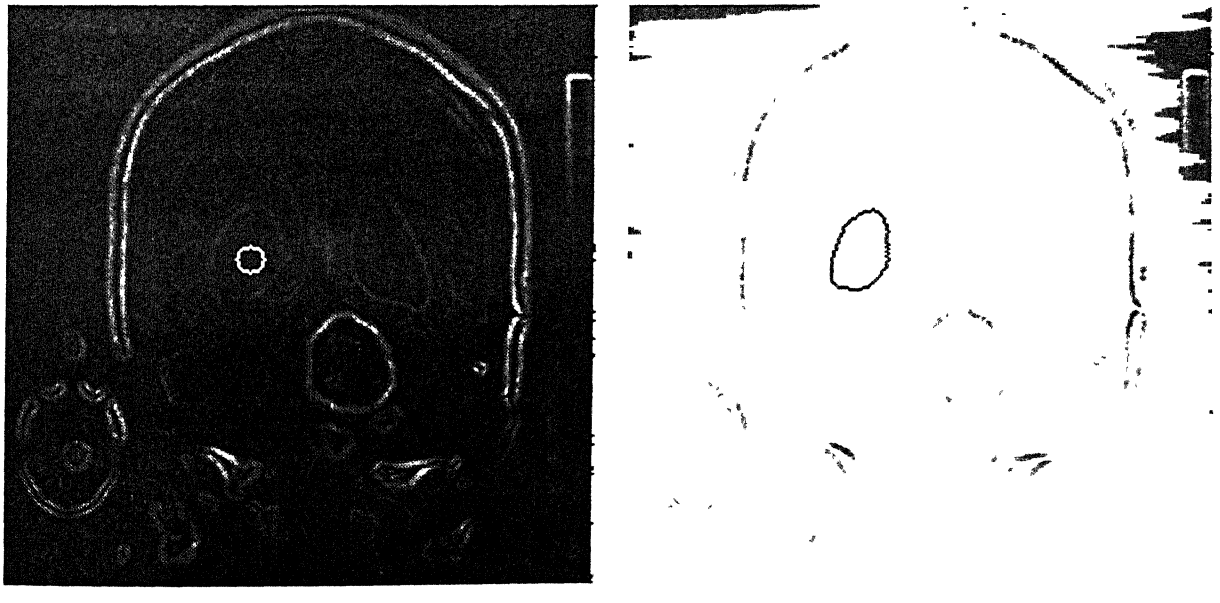
(a) Initial Curve($x_c = 110,\ y_c = 105,\ r = 6$) (b) at step 7

Figure 4.6: Result after 7 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8.$



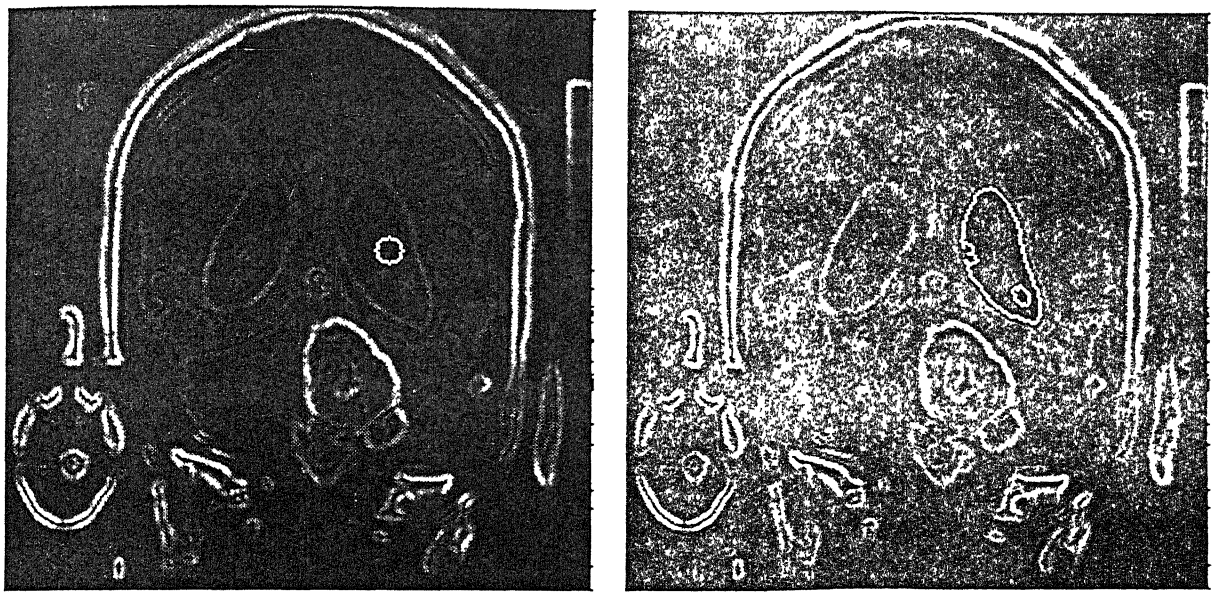
(a) Initial Curve($x_c = 106,\ y_c = 165,\ r = 6$) (b) at step 11

Figure 4.7: Result after 11 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.



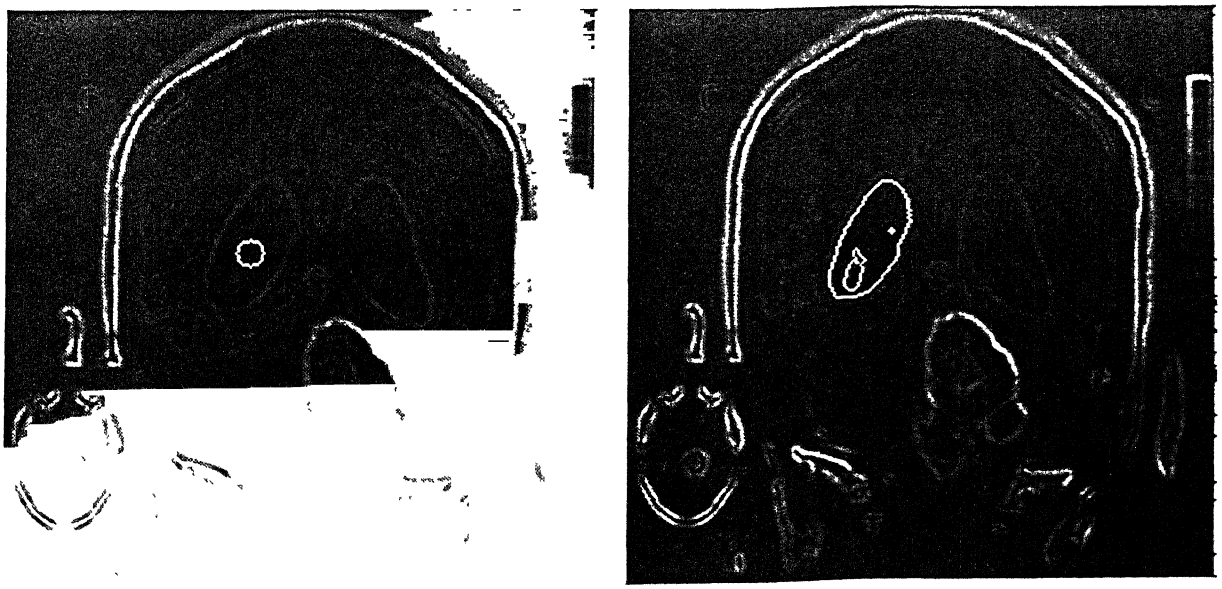
(a) Initial Curve($x_c = 110,\ y_c = 105,\ r = 6$) (b) at step 8

Figure 4.8: Result after 8 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.



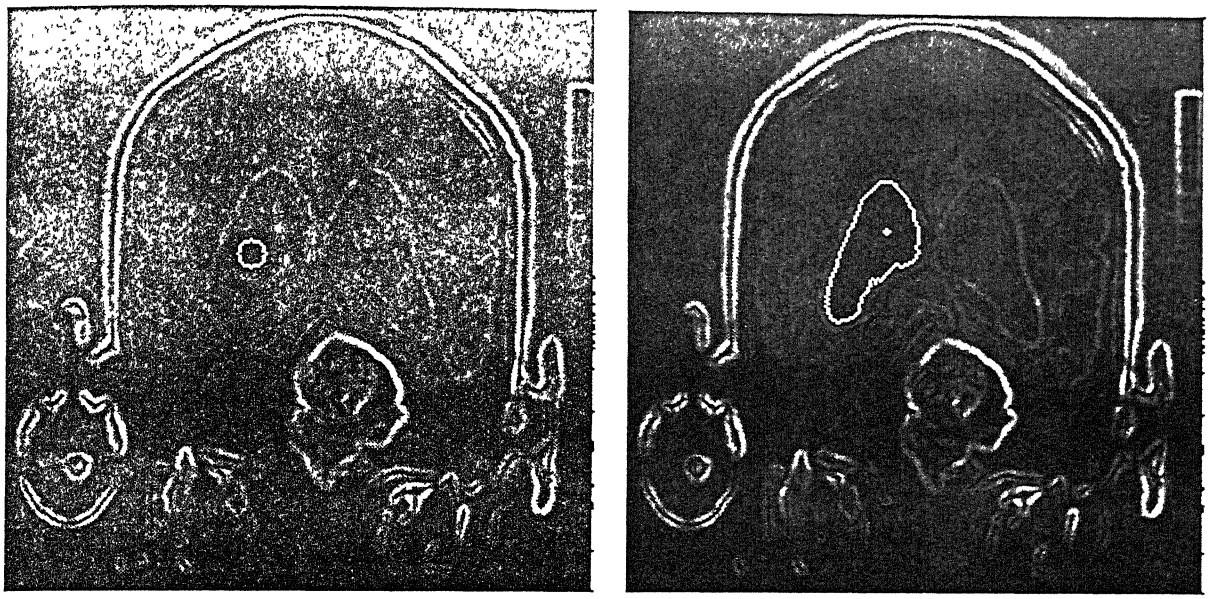
(a) Initial Curve($x_c = 106,\ y_c = 164,\ r = 6$) (b) at step 20

Figure 4.9: Result after 20 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8.$



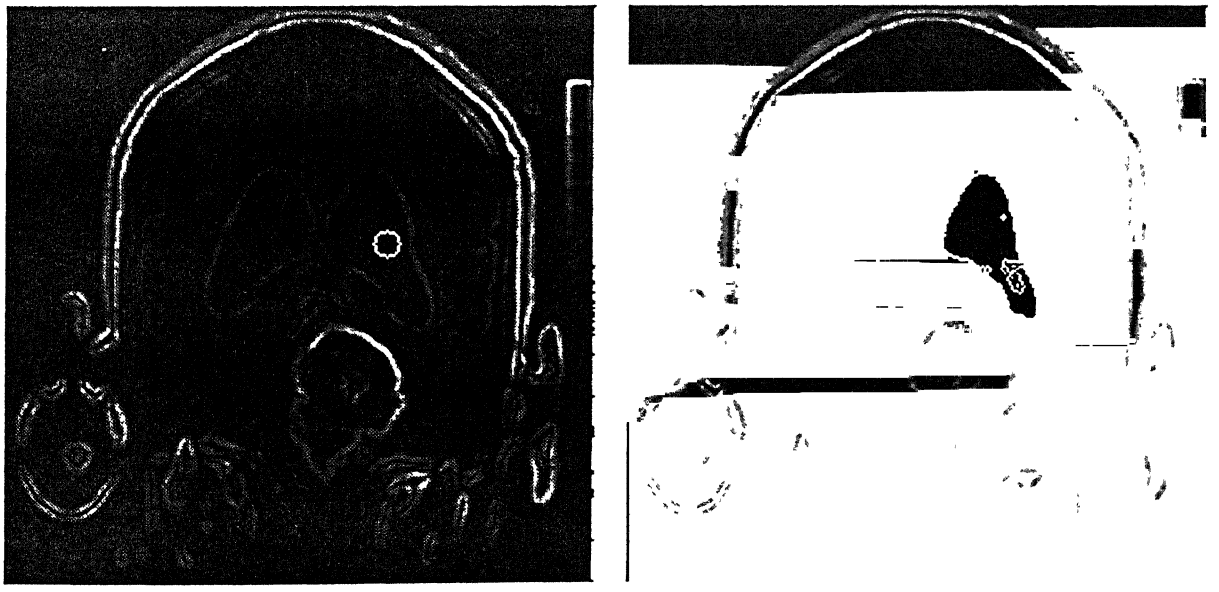
(a) Initial Curve($x_c = 108,\ y_c = 105,\ r = 6$) (b) at step 11

Figure 4.10: Result after 11 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8.$



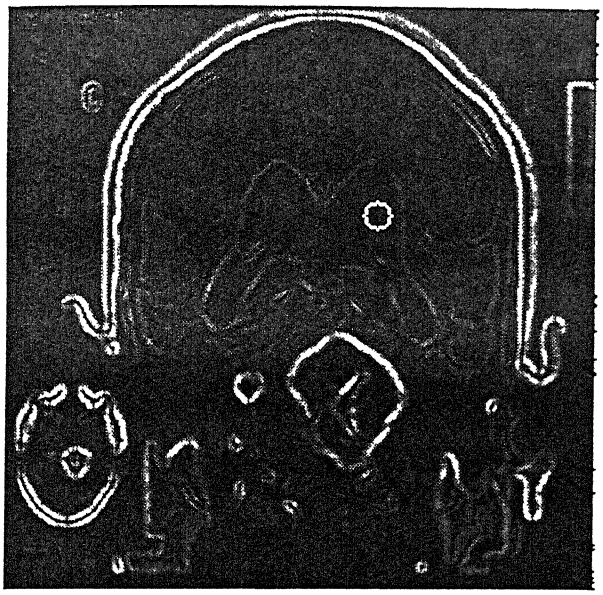
(a) Initial Curve($x_c = 110,\ y_c = 105,\ r = 6$) (b) at step 12

Figure 4.11: Result after 12 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.

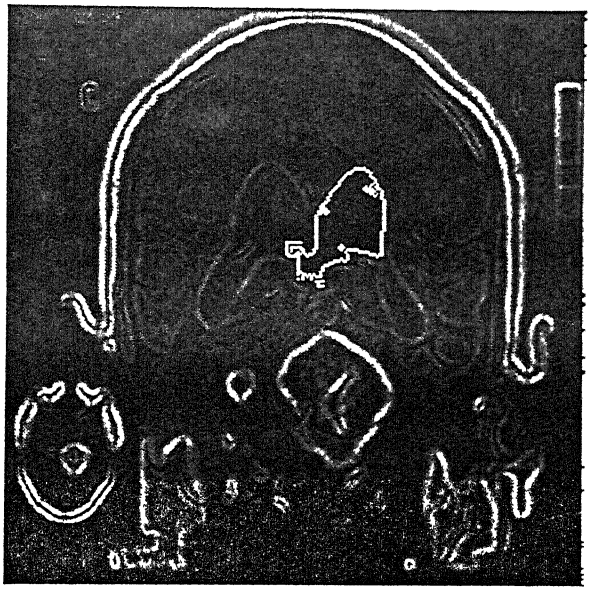


(a) Initial Curve($x_c = 106,\ y_c = 165,\ r = 6$) (b) at step 16

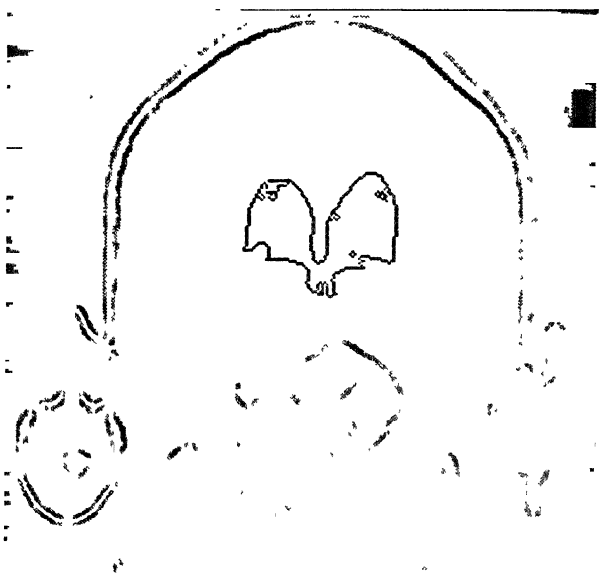
Figure 4.12: Result after 16 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.



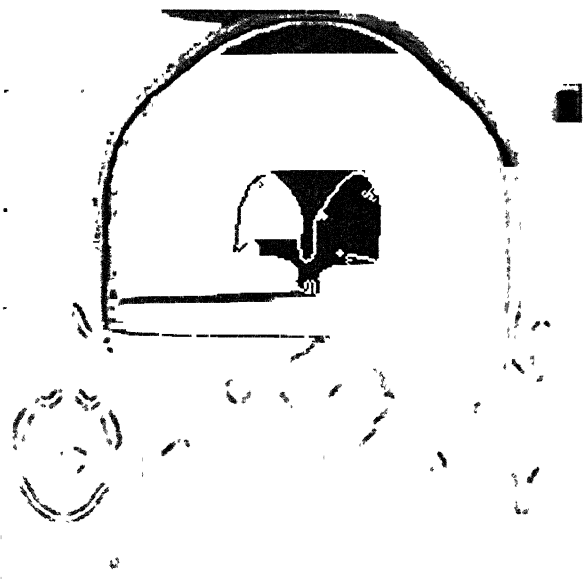
(a) Initial Curve($x_c = 106,\ y_c = 165,\ r = 6$)



(b) at step 10

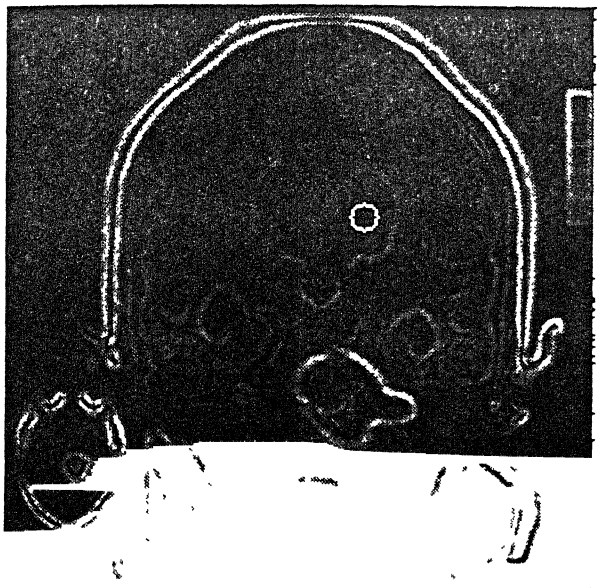


(c) at step 20

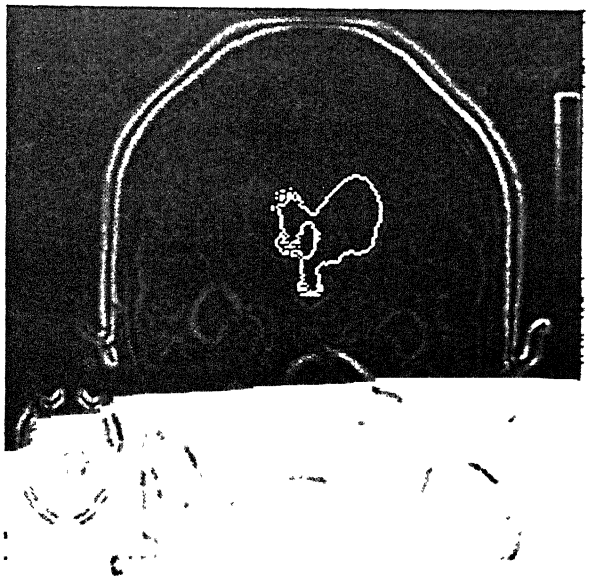


(d) at step 24

Figure 4.13: Result after 24 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.



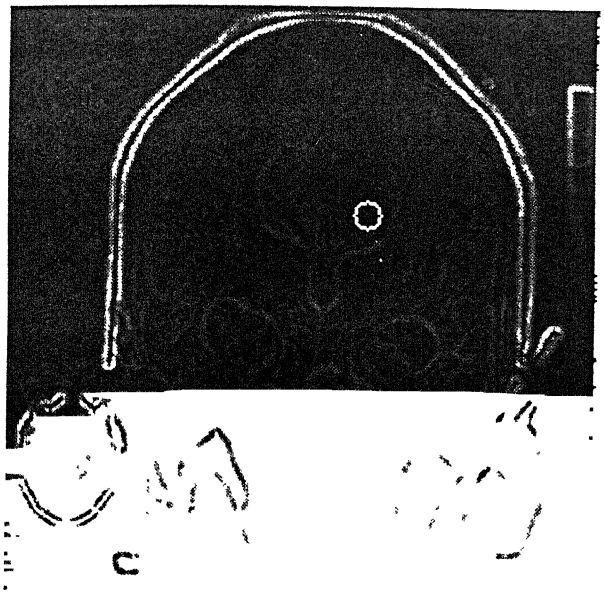
(a) Initial Curve($x_c = 92,\ y_c = 159,\ r = 6$)



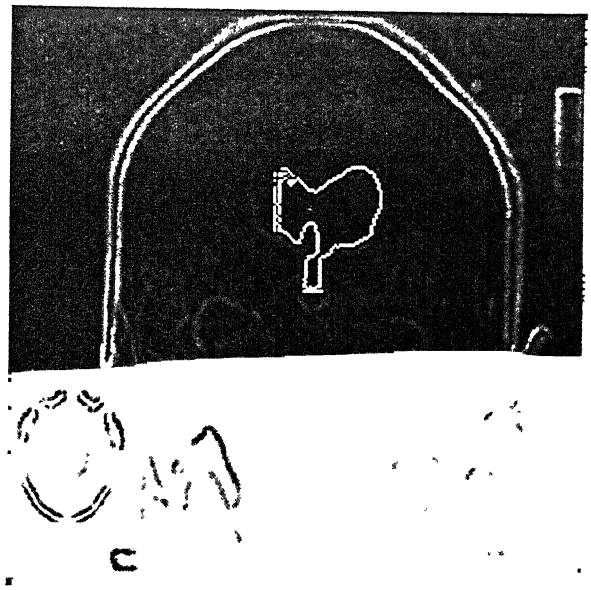
(b) at step 10

(c) at step 21

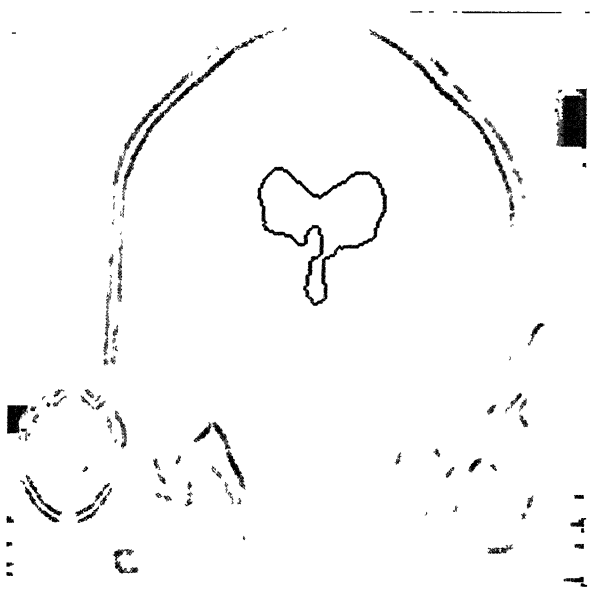
Figure 4.14: Result after 21 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8.$



(a) Initial Curve($x_c = 91,\ y_c = 154,\ r = 6$)



(b) at step 10



(c) at step 16

Figure 4.15: Result after 16 steps with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$.

3D Results

Figure 4.16: An initial closed surface with the parameters $dt = 0.12,\ \epsilon = 0.5,\ \delta = 8$ is placed in the stack of volume

In 3D it is a little bit different from the earlier 2D case. Here we have initialized the surface as a sphere of radius 3 centered at $x_c = 111, y_c = 165, z_c = 3$. The convention of the z_c is nothing but the slice no in stack of slices. As we have started

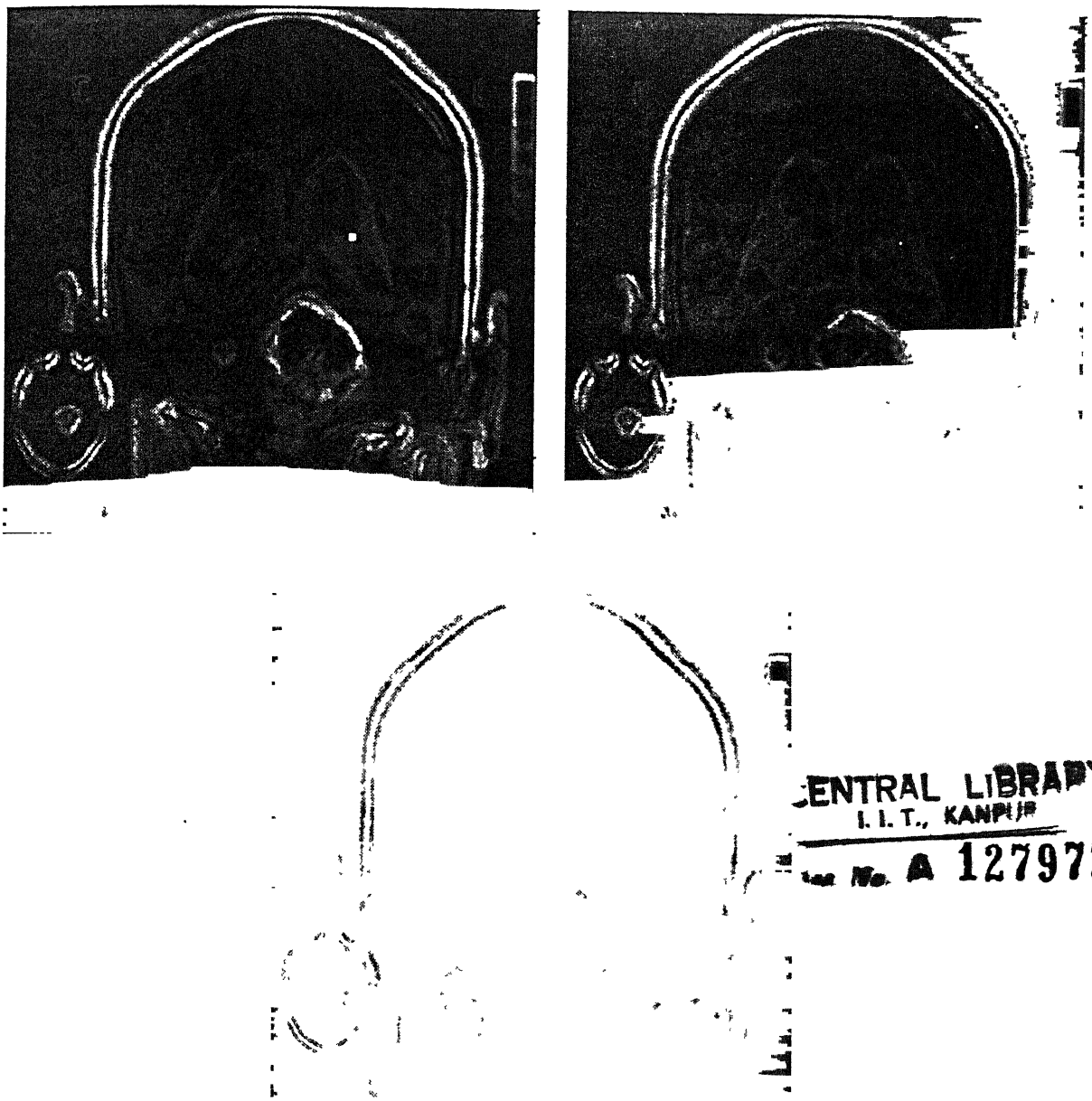
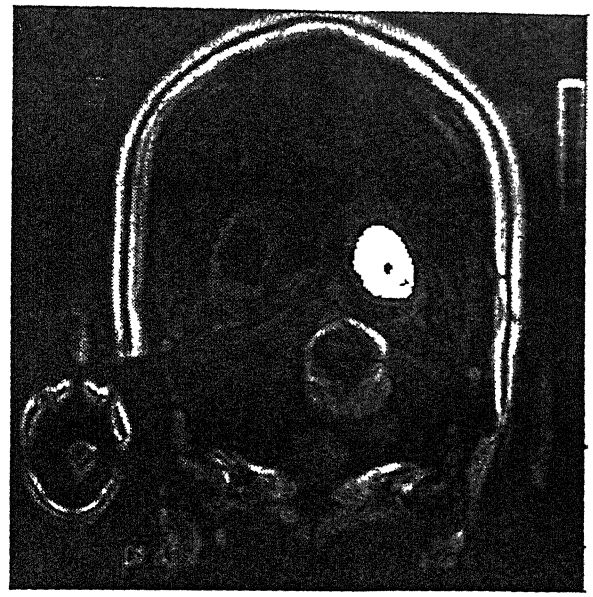
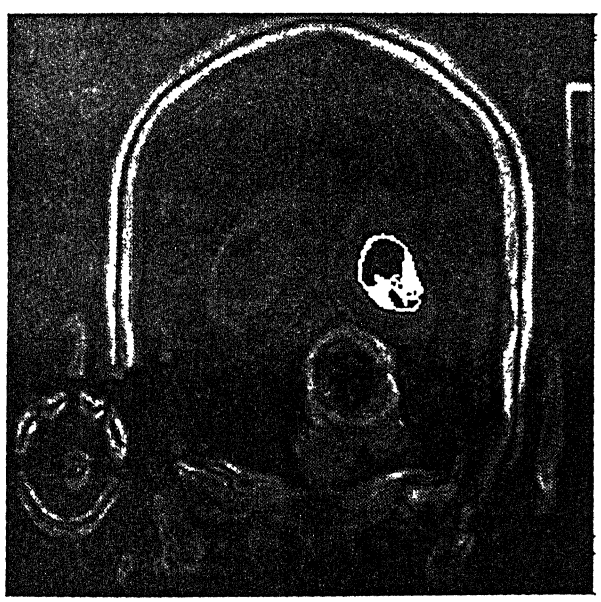


Figure 4.16: An initial closed surface with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$. (*contd.*)

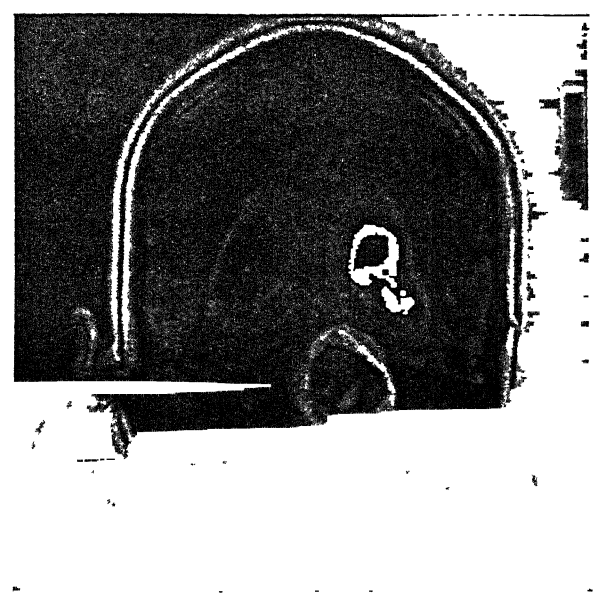
counting zero so 3 means the center is in 4th slice. As we have represented here 3D images as 2D slices so we are able to show the cross section of the sphere cut parallel to the $x - y$ plane. That is why in some of these above figures, there are only circles have been shown .



(a) Slice 1



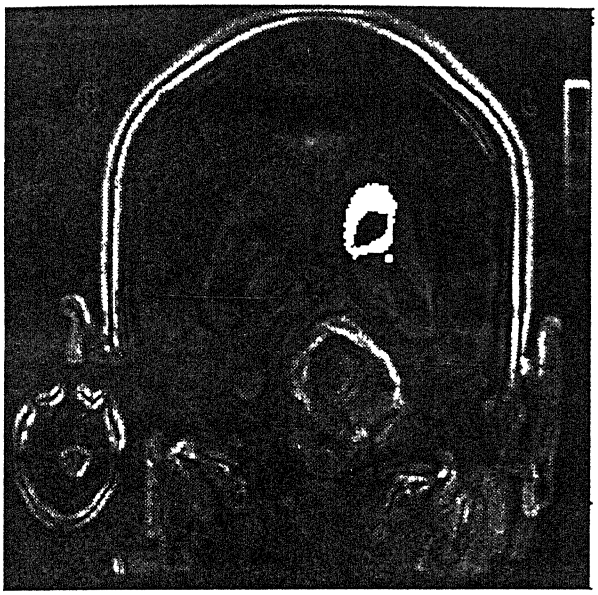
(b) Slice 2



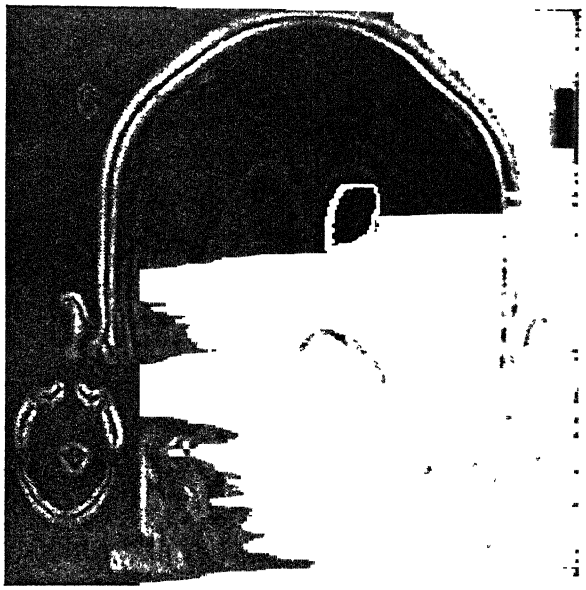
(c) Slice 3

Figure 4.17: After 10 steps the initial surface evolved like this

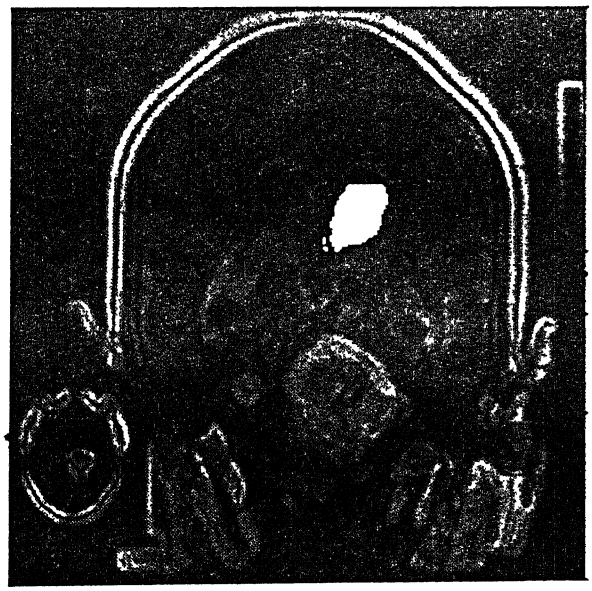
Here the results are shown after 10 steps. In top two slices we can see the surfaces as a top view.



(d) Slice 4

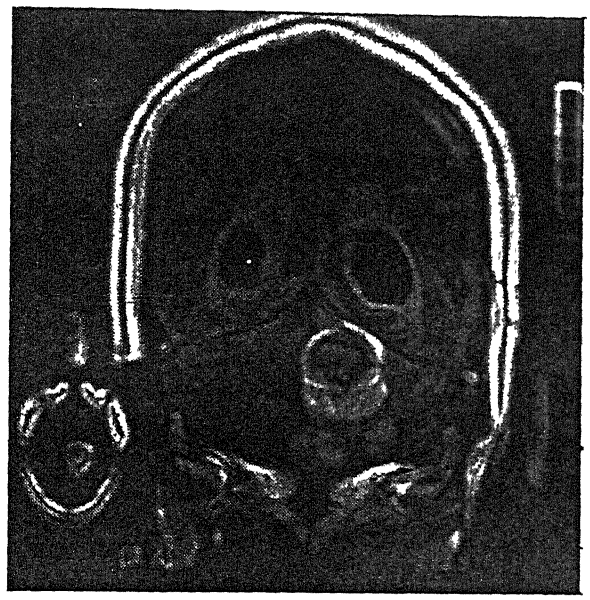


(e) Slice 5

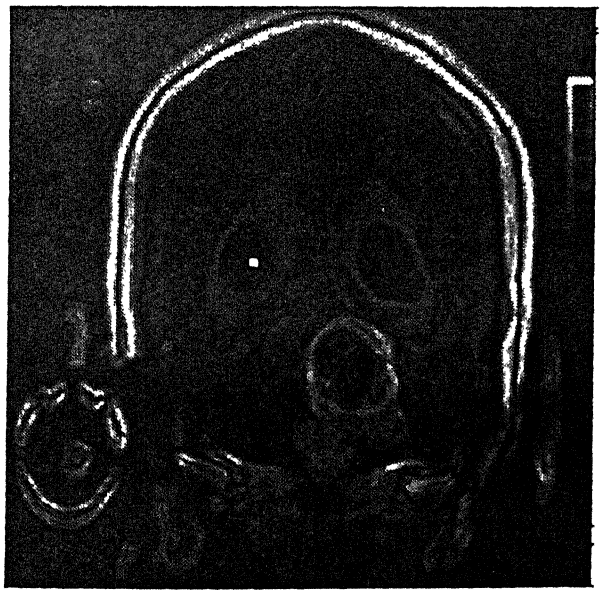


(f) Slice 6

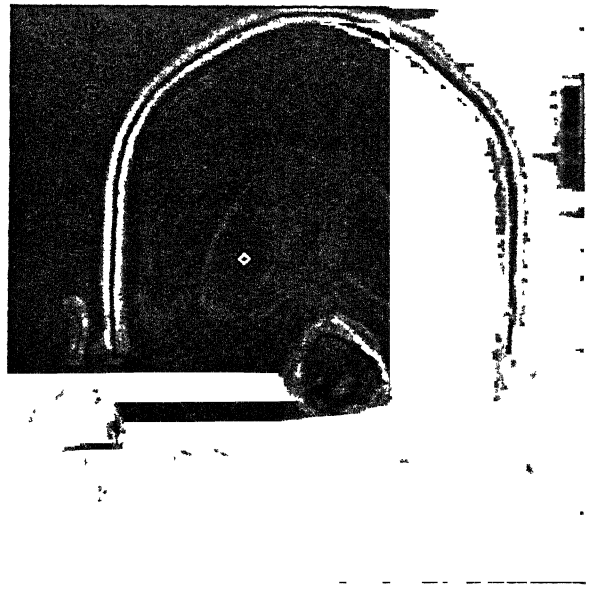
Figure 4.17: After 10 steps the initial surface evolved like this (*contd.*)



(a) Slice 1



(b) Slice 2



(c) Slice 3

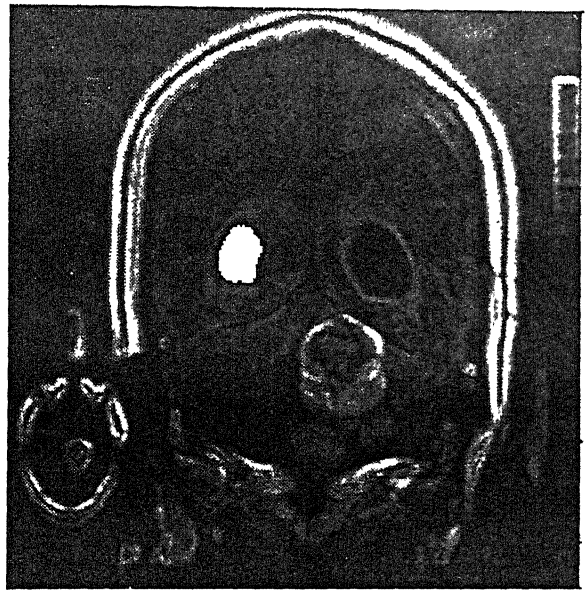
Figure 4.18: An initial closed surface (sphere of radius 3 and center at $x_c = 110, y_c = 100, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$.

(d) Slice 5

(e) Slice 6

(f) Slice 7

Figure 4.18: An initial closed surface (sphere of radius 3 and center at $x_c = 110, y_c = 100, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$. (*contd.*)

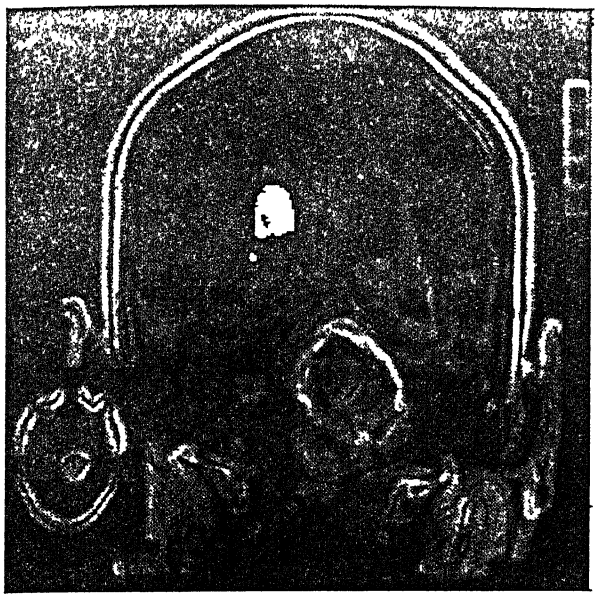


(a) Slice 1

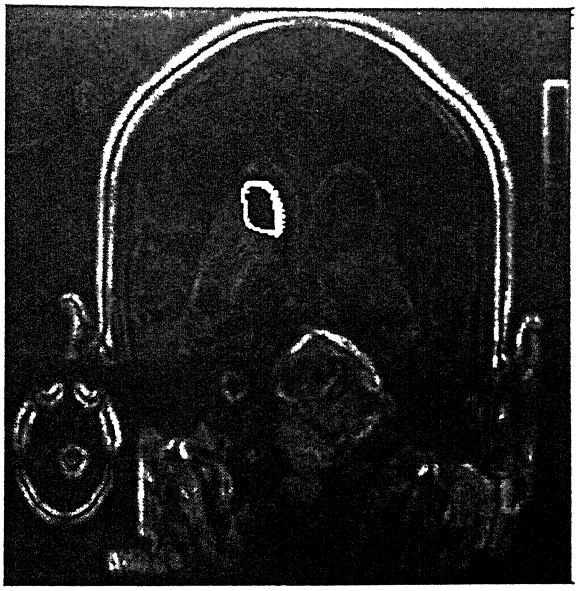
(b) Slice 2

(c) Slice 3

Figure 4.19: After 10 steps the initial surface evolved like this



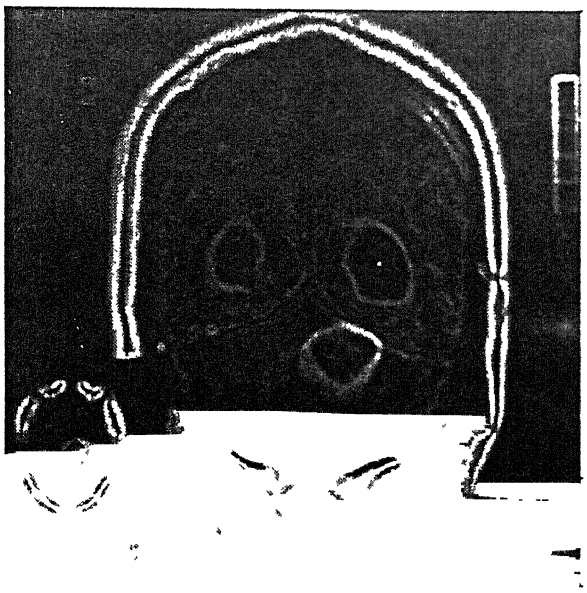
(d) Slice 4



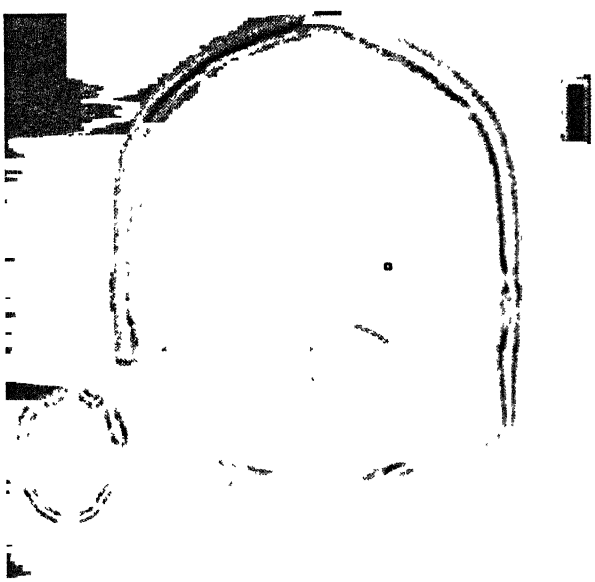
(e) Slice 5

(f) Slice 6

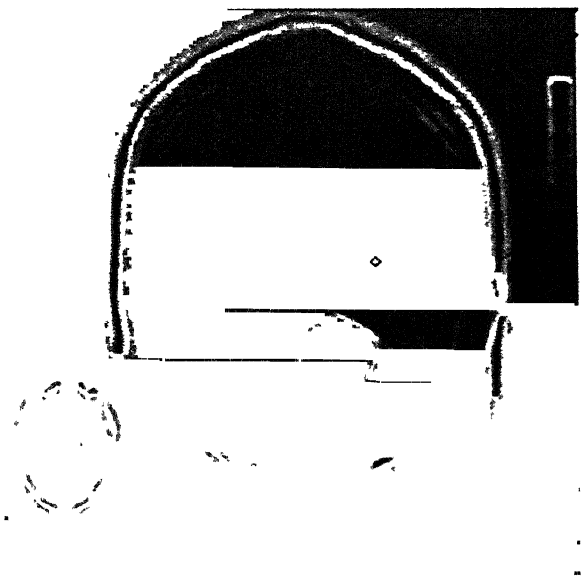
Figure 4.19: After 10 steps the initial surface evolved like this (*contd.*)



(a) Slice 1

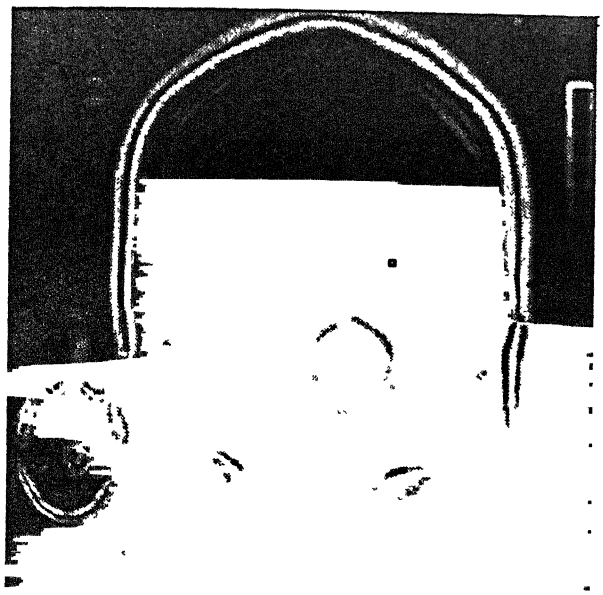


(b) Slice 2

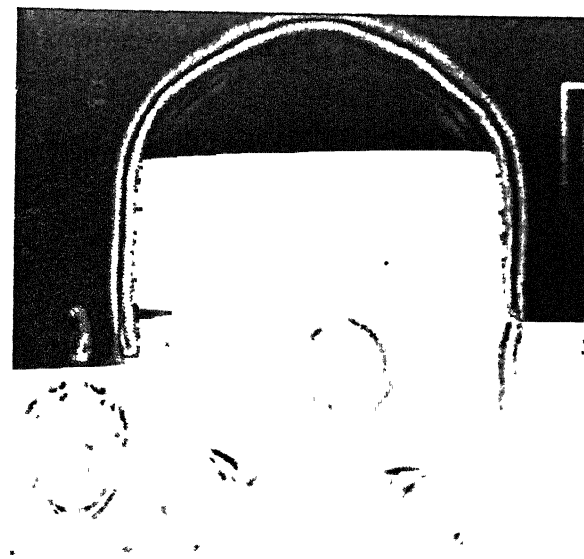


(c) Slice 3

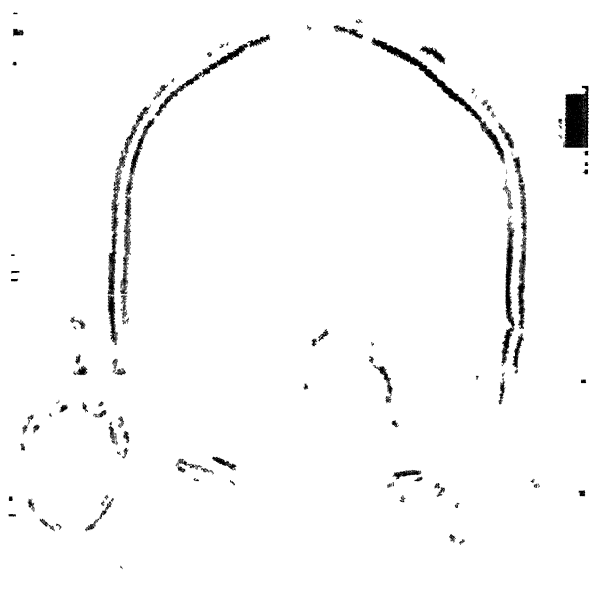
Figure 4.20: An initial closed surface (sphere of radius 3 and center at $x_c = 111, y_c = 165, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$.



(d) Slice 5

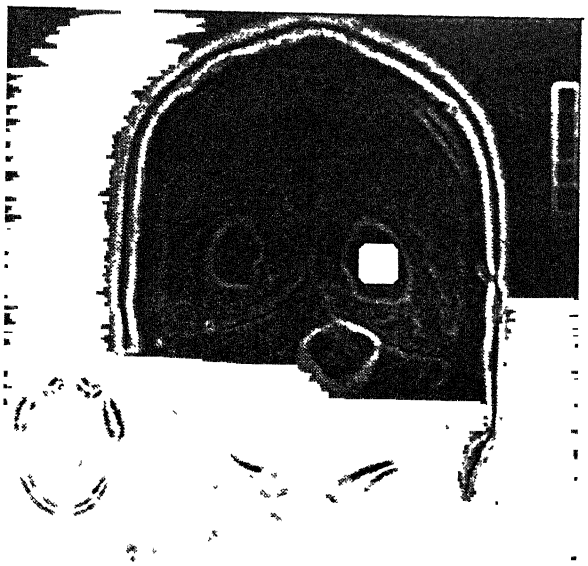


(e) Slice 6

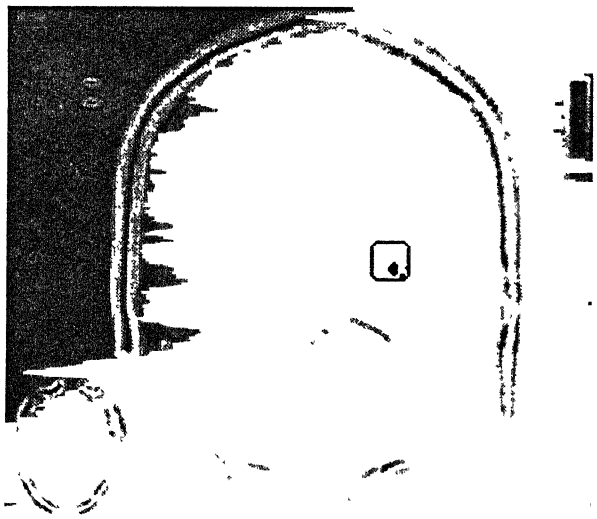


(f) Slice 7

Figure 4.20: An initial closed surface (sphere of radius 3 and center at $x_c = 111, y_c = 165, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$. (*contd.*)



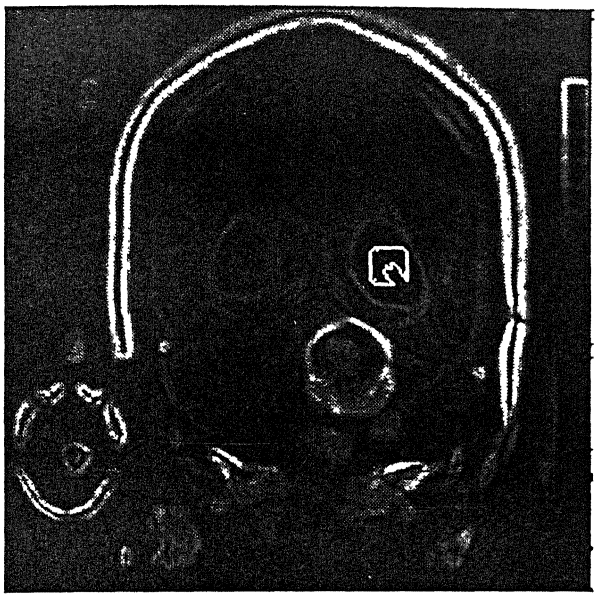
(a) Slice 1



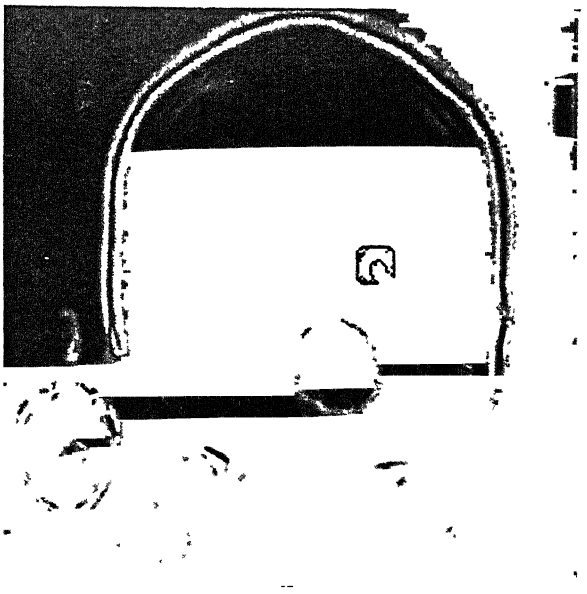
(b) Slice 2

(c) Slice 3

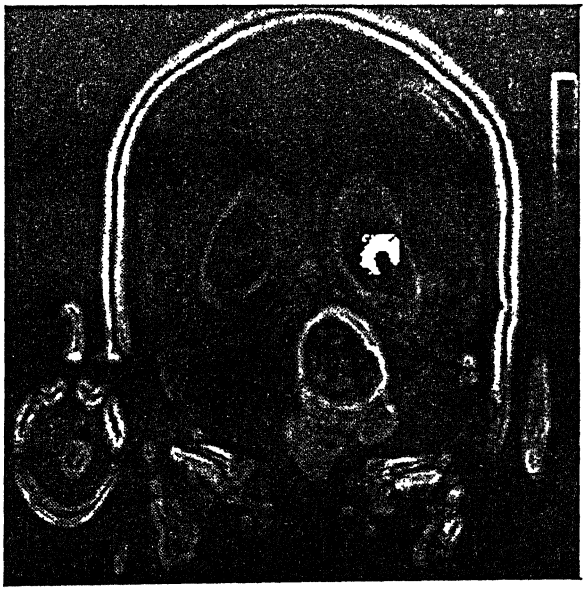
Figure 4.21: After 2 steps the initial surface evolved like this.



(d) Slice 4

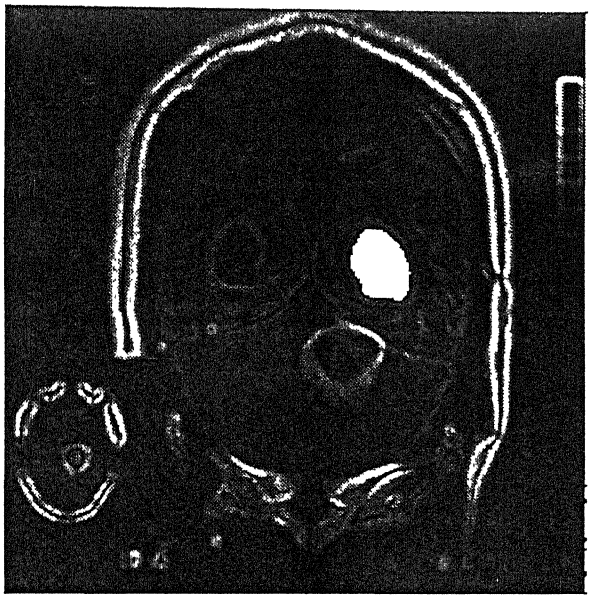


(e) Slice 5

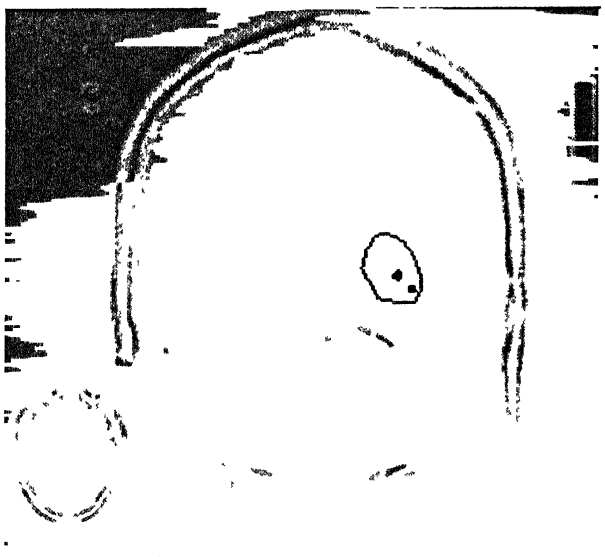


(f) Slice 6

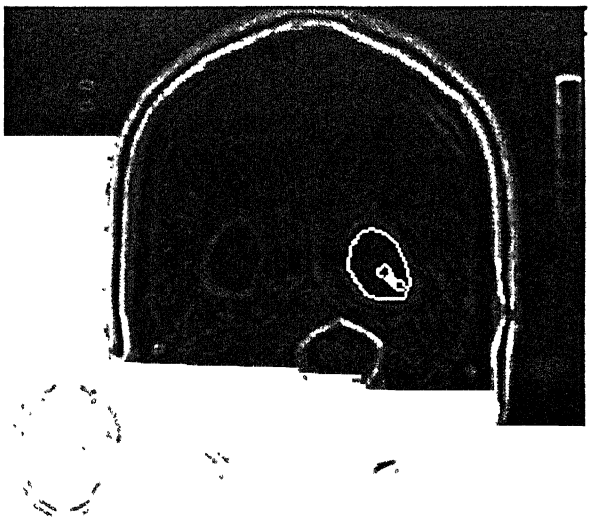
Figure 4.21: After 2 steps the initial surface evolved like this. (*contd.*)



(a) Slice 1

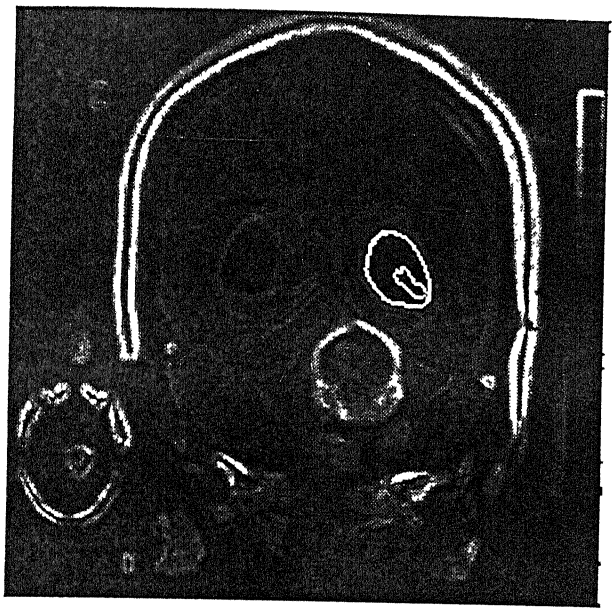


(b) Slice 2

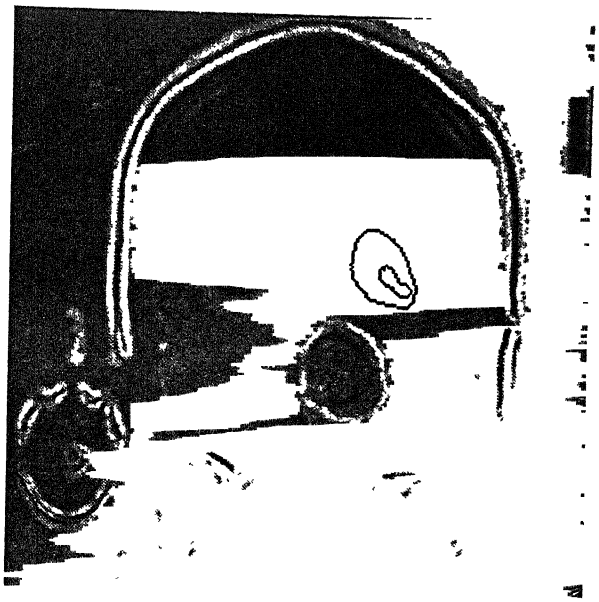


(c) Slice 3

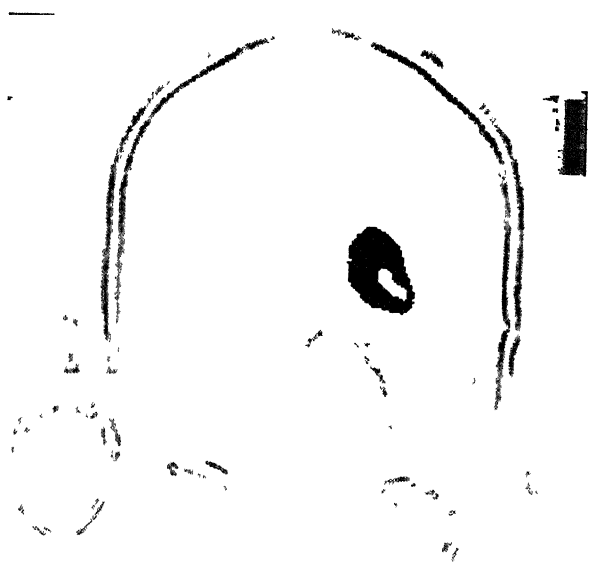
Figure 4.22: After 9 steps the initial surface evolved like this



(d) Slice 4

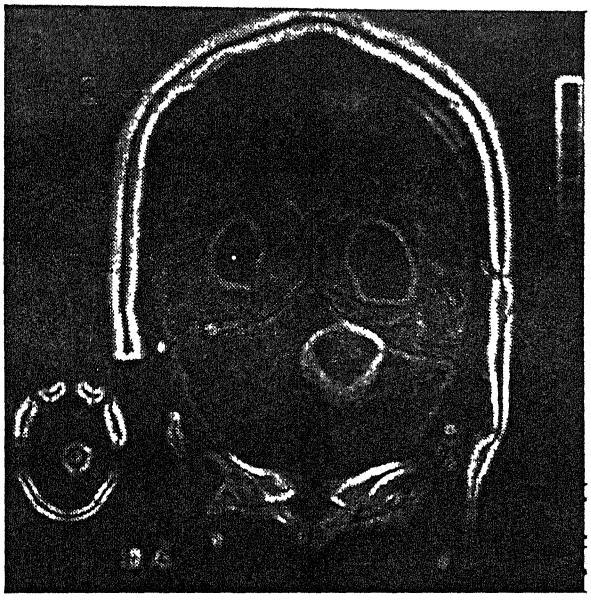


(e) Slice 5

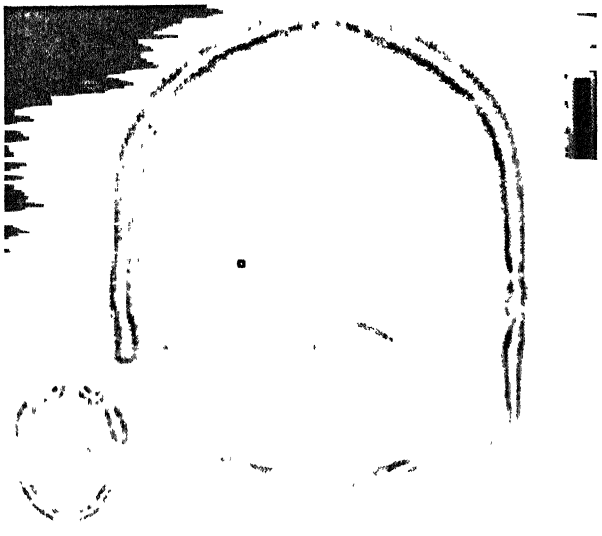


(f) Slice 6

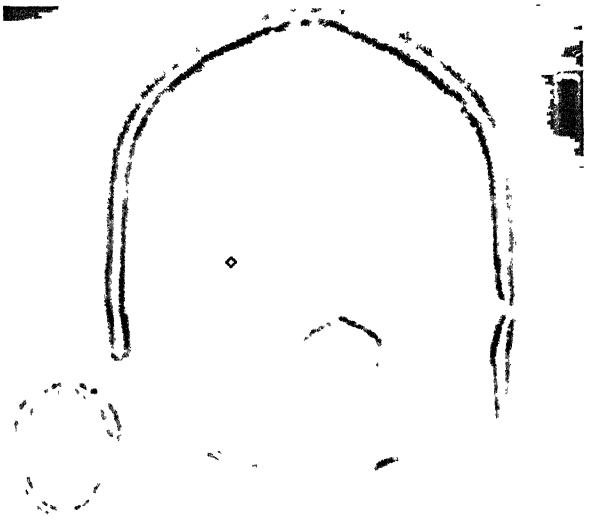
Figure 4.22: After 9 steps the initial surface evolved like this (*contd.*)



(a) Slice 1

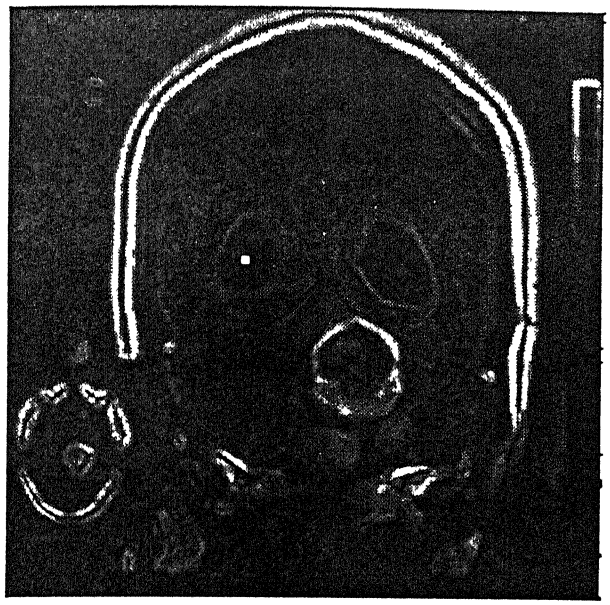


(b) Slice 2

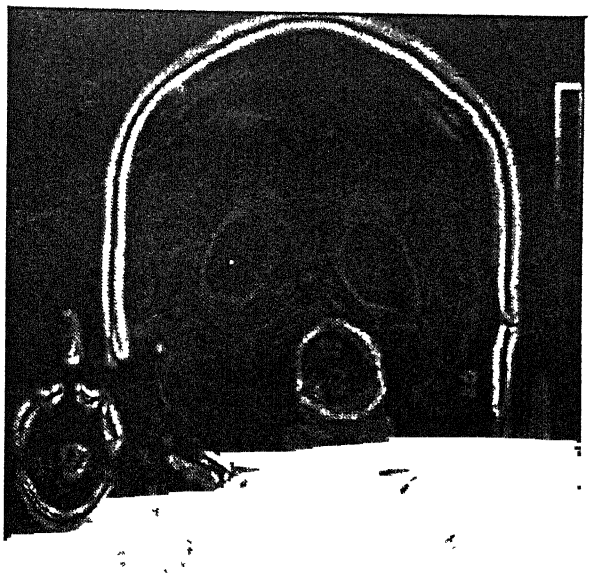


(c) Slice 3

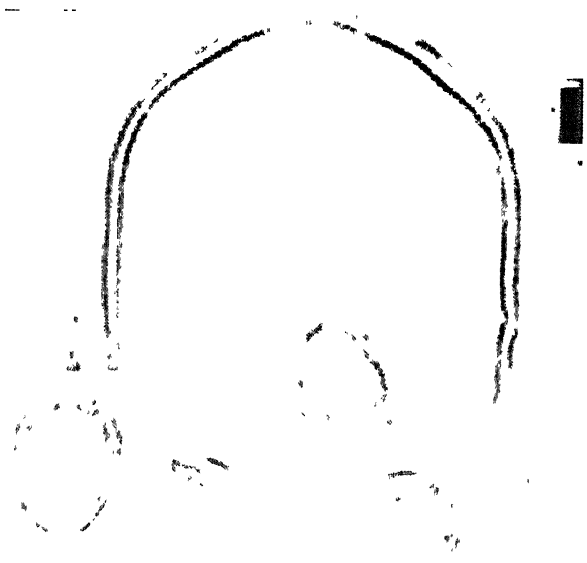
Figure 4.23: An initial closed surface (sphere of radius 3 and center at $x_c = 110, y_c = 100, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$.



(d) Slice 4

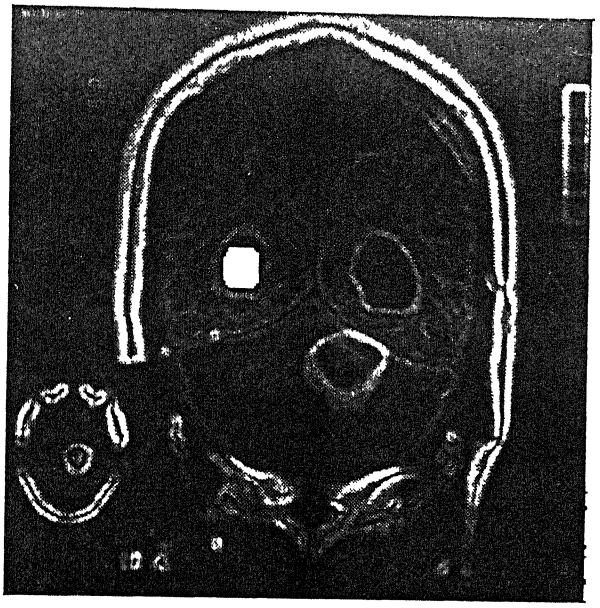


(e) Slice 5

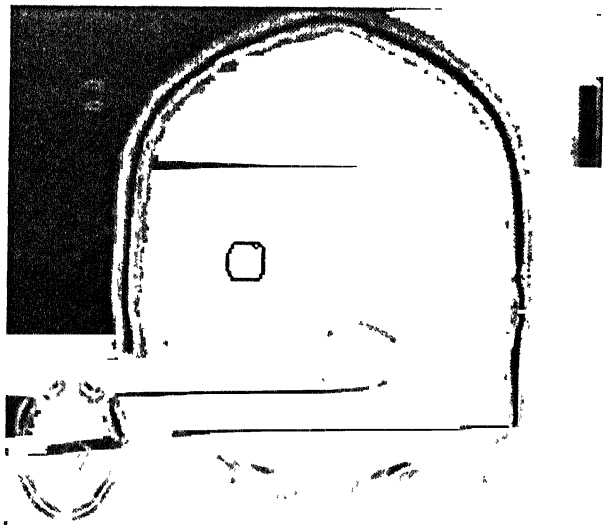


(f) Slice 6

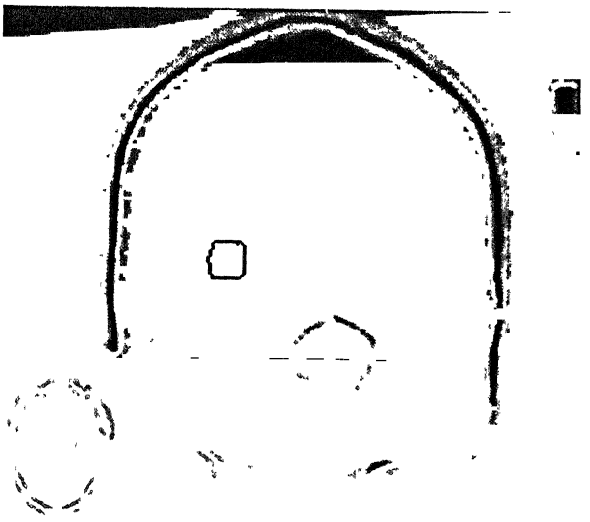
Figure 4.23: An initial closed surface (sphere of radius 3 and center at $x_c = 110, y_c = 100, z_c = 3$) is placed in these stack of volume with the parameters $dt = 0.12$, $\epsilon = 0.5$, $\delta = 8$. (*contd.*)



(a) Slice 1

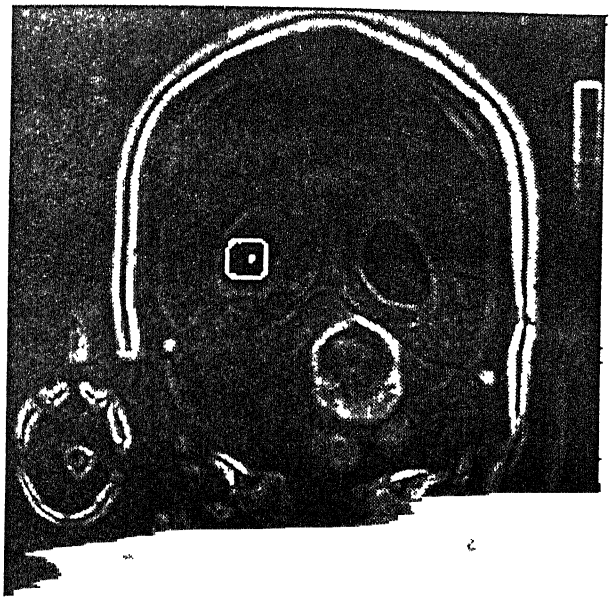


(b) Slice 2

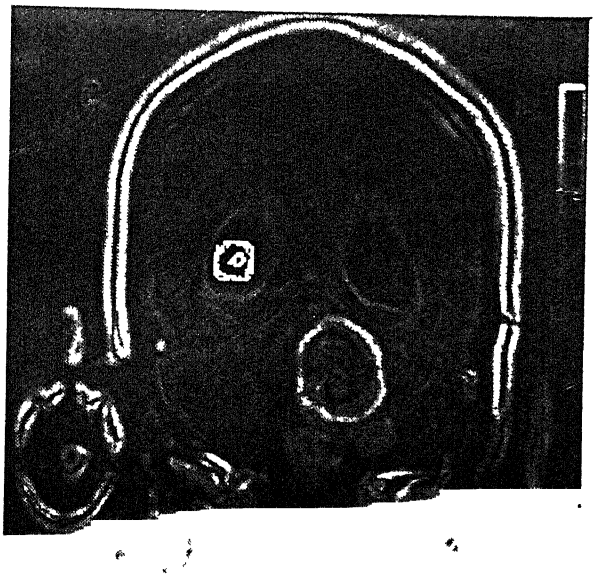


(c) Slice 3

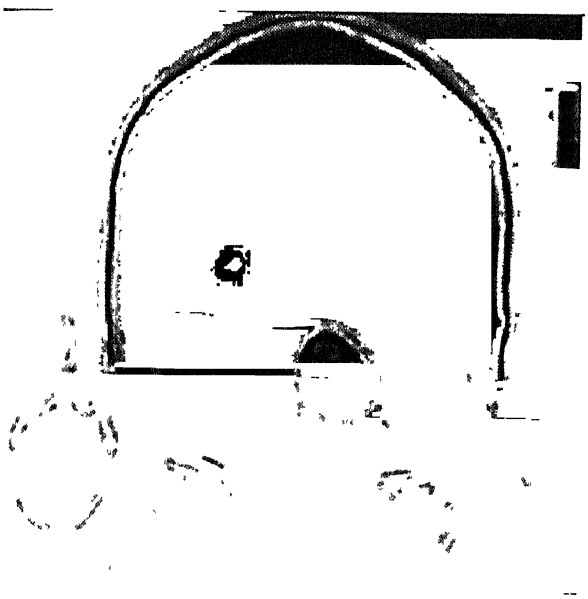
Figure 4.24: After 2 steps the initial surface evolved like this.



(d) Slice 4

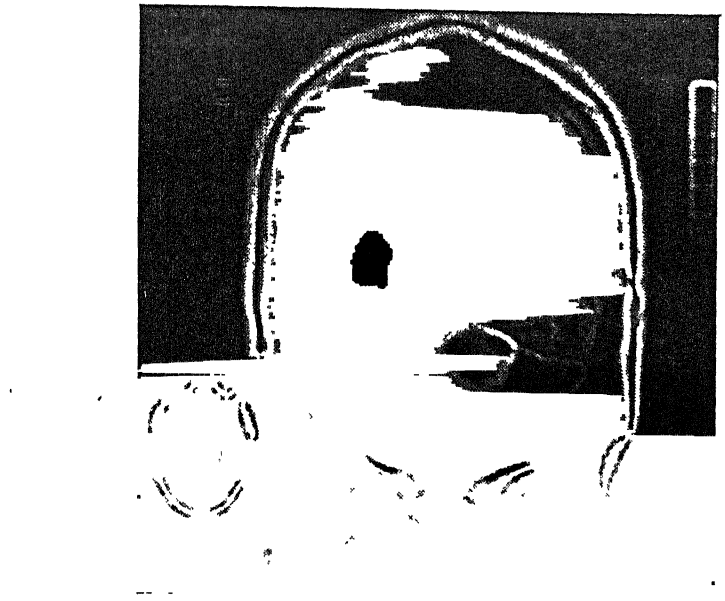


(e) Slice 5

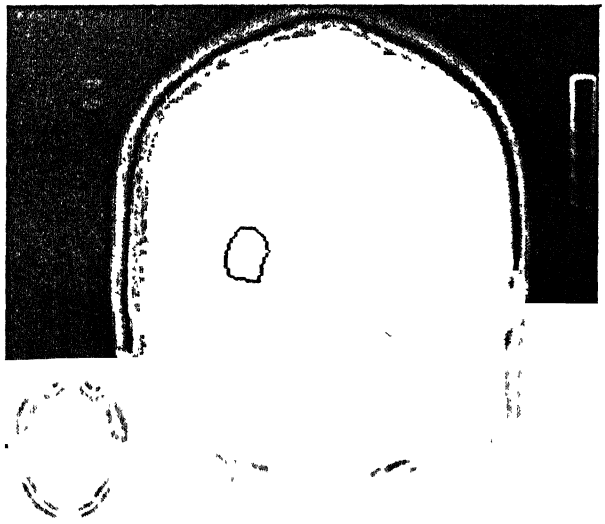


(f) Slice 6

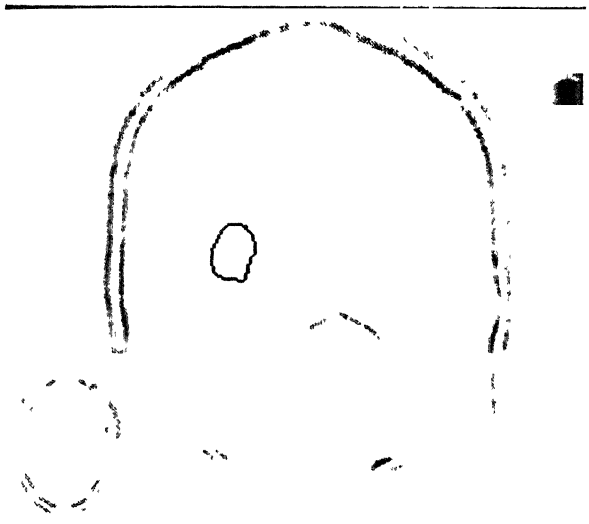
Figure 4.24: After 2 steps the initial surface evolved like this. (*contd.*)



(a) Slice 1

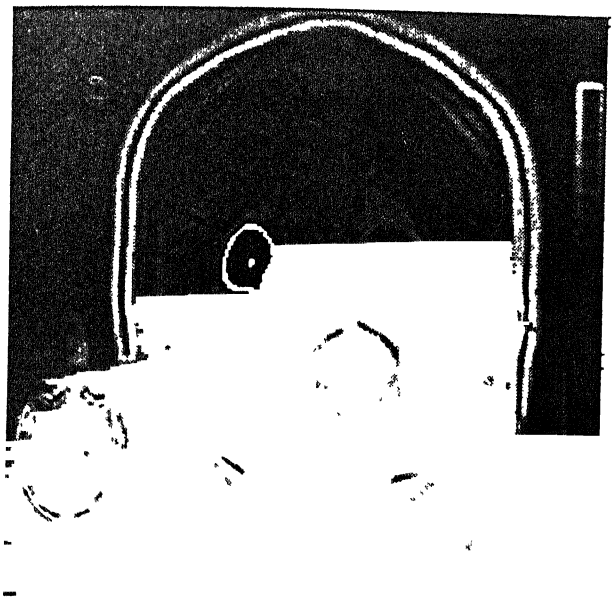


(b) Slice 2

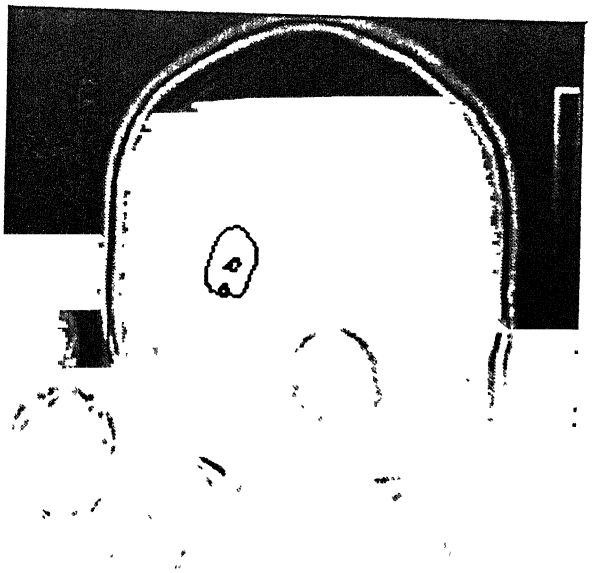


(c) Slice 3

Figure 4.25: After 9 steps the initial surface evolved like this.



(d) Slice 4



(e) Slice 5

(f) Slice 6

Figure 4.25: After 9 steps the initial surface evolved like this. (*contd.*)

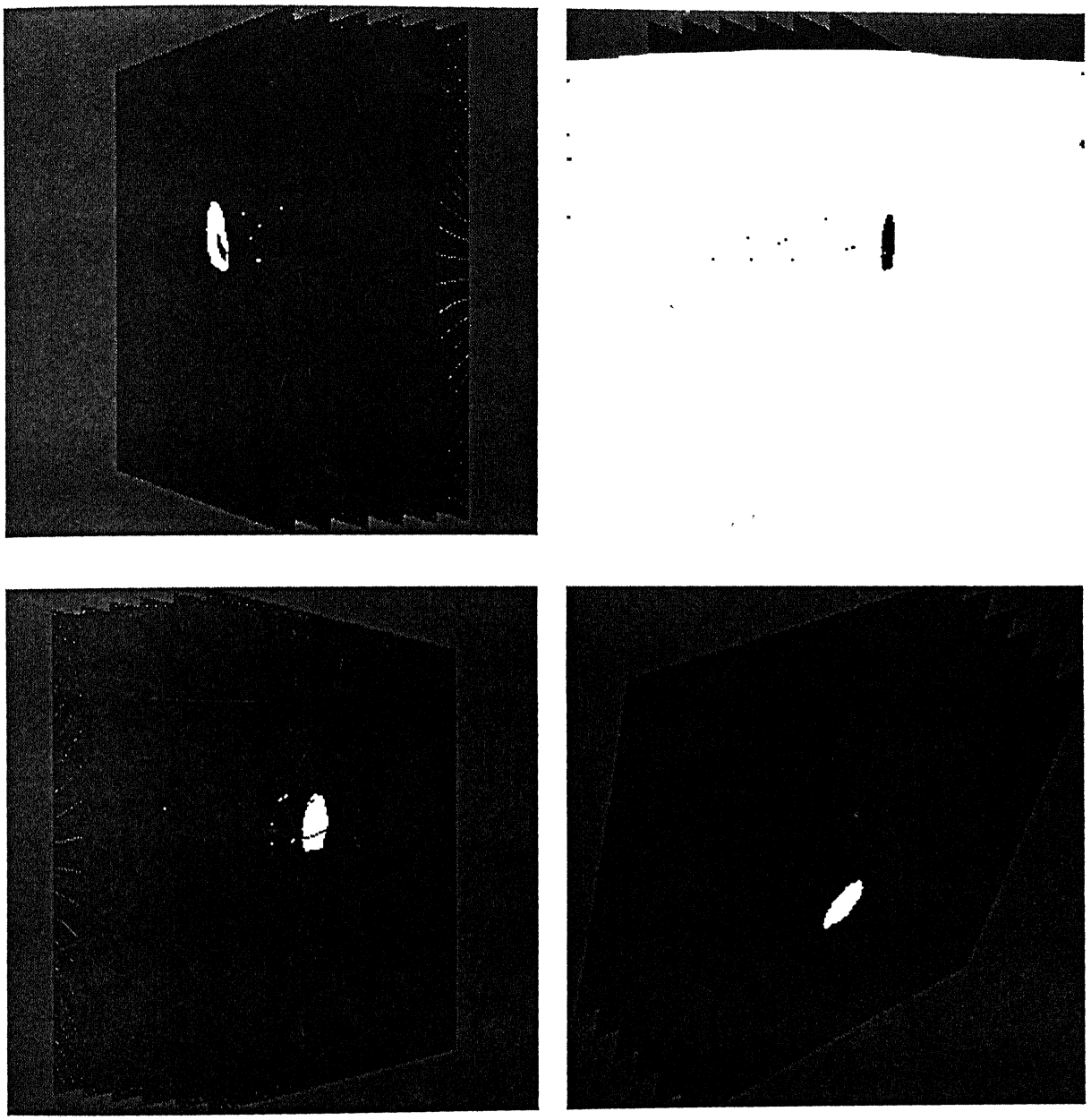


Figure 4.26: Four Slices of the segmented image shown using VTK.

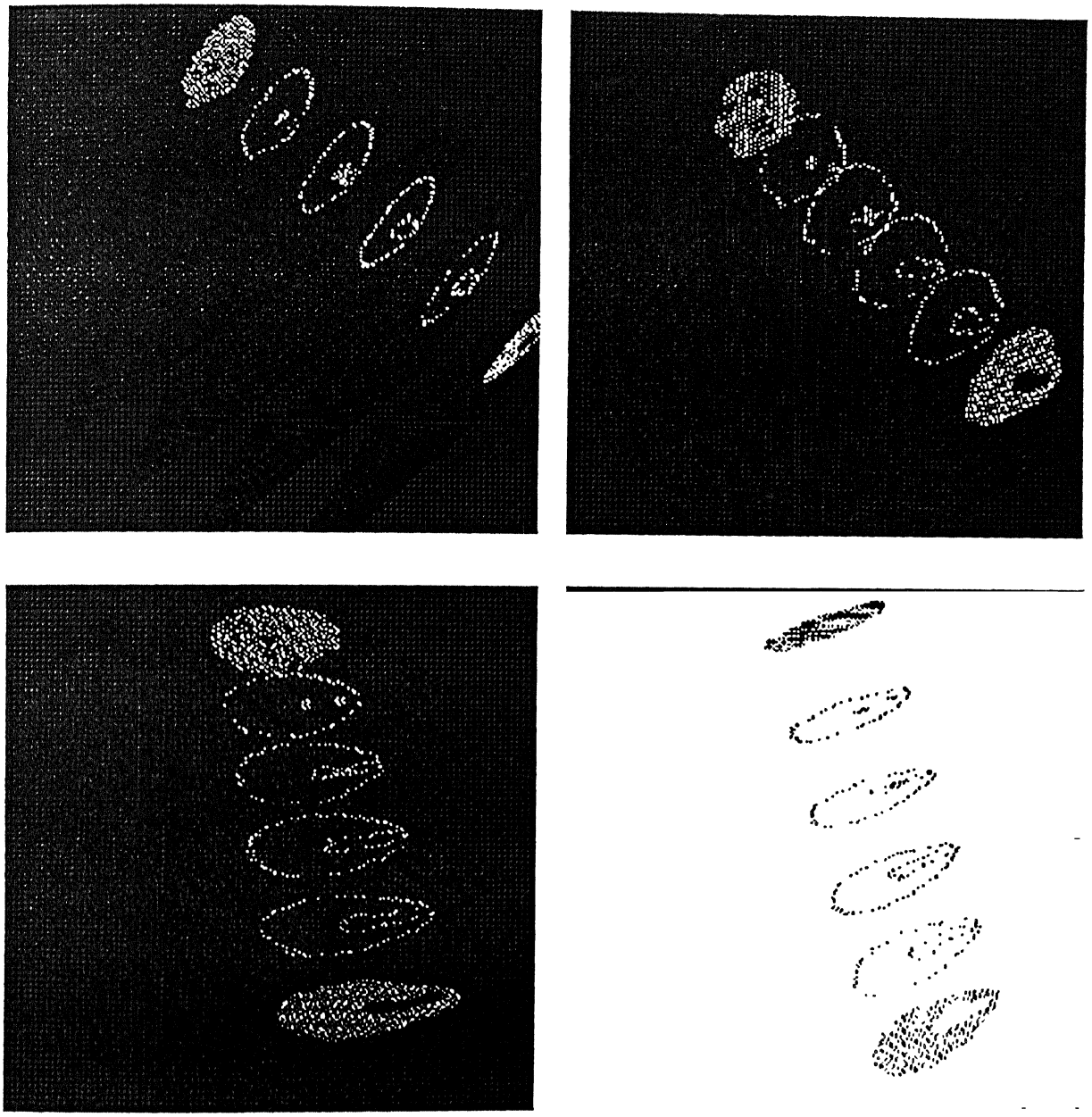


Figure 4.27: Four Slices of the segmented image using VTK, here the VOI (volume of interest) is different

4.2 Discussion

As it can be seen from section 4.1 the method is giving the accurate boundary of the object of interest. As in other type of algorithm of segmentation one has to give more effort to get the boundary of a particular object. In 2D results, in Fig 4.2 , 4.3 and 4.4 we have shown three different regions with intermediate steps. The background of each output result is being set equal to the gradient of the input image for a visual clarity. We have achieved this by writing the input image gradient and the segmented image into the same image plane so that we don't have to specify input image each time. The time step taken for getting the final segmented image depends upon many things. As shown on those figures the result have been achieved in 4 steps each consisting 50 inner iteration. We have taken the radius of the initial circle as 6. If we take the initial radius as 3 then it takes around 7 steps before coming to a complete stop. Also we can control the smoothness of the snake by changing the value of ϵ . In Fig 4.5 to Fig 4.15 only the initial curve and the final curve in different slices are shown.

In 3D results, Fig 4.16 shows the 8 slices on which the initialization has been done i.e. the initial sphere has been kept in . After 10 steps the results of this 3D segmentation was shown in the Fig 4.17. It is very clear from the images that the results are not good as expected. We observed that as the algorithm treats all the points as a cluster of points, so it does not differentiate between planes. As the inter pixel distance in $x-y$ plane is different from both $y-z$ and $x-z$ plane so it leads to detecting the boundary of unwanted objects. As the 3D surface evolves in all the 3 directions so unwanted surface was being built from the other slices i.e.the initial surface from the other slices disturbs the accurate boundary detection. To mitigate this problem we have interpolated some slices in between. Because of the interpolation the continuity in 3D is maintained. The results of interpolation have been shown in Fig 4.20 to Fig 4.25 with intermediate steps.

In the last two figures we have used VTK for the visualization of 5 slices. Apart from rotating the figure in any direction we are also able to zoom it to different scale.

4.3 Conclusion

We have implemented the Level Set theories with a straight forward extension of the speed function. Though our method uses the concept of snake but we have successfully overcome the problem of change in topology of the evolving front. In the classical snake method one requires an initial curve which should be very close to the exact boundary, so one needs to be careful in the choice of initial curve. In our method there are no such restrictions for the initial curve. Thus We have achieved both flexibility and economy in time as in real time our method hardly takes 30 seconds in 2D and 60 seconds in 3D (in a pentium II)for boundary detection. Our method is also user friendly as it requires less expertise to use.

4.4 Future Work

As we have mentioned the extended speed function which has a powerful stopping criterion property can be implemented to get better results in 3D. As it is computationally expensive for the computer used for this simulation we were unable to implement this. The calculation of the volume of the object of interest can also be taken as a subject of further studies.

Bibliography

[1] Kass, M., Witkin, A., and Terzopoulos, D., *Snakes: Active Contor Models,* International J. of Computer Vision 1, pp. 321-331, 1988.

[2] Gonzalez, C.R., and Woods, E.R., *Digital Image Processing.*

[3] Sethian, J.A., *Level Set Methods*, University of California, Berkeley.

[4] Clark, C.M., Hall, O.L., Goldgof, D.B., Clarke, L.P., Velthuizen R.P., Silbiger S.M., *MRI Segmentation using Fuzzy Clustering Techniques.* IEEE Engineering in Medicine and Biology, pp. 730-740, November/December 1994.

[5] Clark, C.M., Hall, O.L., Goldgof, D.B., Velthuizen R.P., Silbiger S.M., Murtagh, F.R., *Automatic Tumor Segmentation using Knowledge-based Techniques.* IEEE Trans. on Medical Imaging, vol 17, N0. 2, pp. 187-201, April 1998.

[6] Osher, S., and Sethian, J.A., *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulation,* Journal of Computational Physics, 79, pp. 12-49, 1988.

[7] Noh, W., and Woodward, P., *A Simple Line Interface Calculation.* Proceedings, Fifth International Conference on Fluid Dynamics, Eds. A.I. vn de Vooran and P.J. Zandberger, Springer-Verlag, 1976.

[8] Sethian, J.A., *Parallel Level Set Methods for Propagating Interfaces on the Connection Machine,* Unpublished manuscript, 1989.

[9] Milne, B. *Adaptive Level Set Methods Interfaces*, PhD. Thesis, Dept. of Mathematics, University of California, Berkeley, CA.,1995.

[10] Berger, M., and Colella, P., *Local Adaptive Mesh Refinement for Shock Hydrodynamics,* J. Comp. Phys., 1, 82, pp. 62-84, 1989.

[11] Chopp, D.L., *Computing Minimal Surfaces via Level Set Curvature Flow,* Jour. of Comp. Phys., 106, pp. 77-91, 1993.

[12] Lorenson, W.E., and Cline, H.E., *Marching Cubes: A High Resolution 3D Surface Construction Algorithm,* Computer Graphics, 21, 4, 1987.

[13] Sussman, M., Smereka, P. and Osher, S.J., *A Level Set Method for Computing Solutions to Incompressible Two-Phase Flow,* J. Comp. Phys. 114, pp. 146-159, 1994.

[14] Bourlioux, A., *A Coupled Level-Set Volume of Fluid Algorithm for Tracking Material Interfaces,* Sixth International Symposium on Computational Fluid Dynamics, Sept. 4-8, 1995, Lake Tahoe, NV.

[15] Sethian, J.A., *Algorithms for Tracking Interfaces in CFD and Material Science,* Annual Review of Computational Fluid Mechanics, 1995.

[16] Sethian, J.A., *Curvature Flow and Entropy Condition Applied to Grid Generation*, J. Comp. Phys., 115, pp. 440-454, 1994

[17] Kimmel, R., and Bruckstein, A., *Shape Offsets via Level Sets,* Computer Aided Design, 25, 3, pp. 154-161, 1993.

[18] Malladi, R., Sethian, J., and Vemuri, B.C., *Shape Modelling with Front Propagation: A Level Set Approach*, IEEE Trans. on PAMI, Vol 17, No. 2, pp. 158-174, 1995.

[19] Bourlioux, A., and Sethian, J.A., *Projection Methods Coupled to Level Set Interface Methods*, to be submitted, Journal of Comp. Phys., 1994.